Escola Universitaria Politécnica

UNIVERSIDADE DA CORUÑA

Grado en Ingeniería Electrónica Industrial y Automática

TRABAJO DE FIN DE GRADO

TFG Nº: **770G01A162**

TÍTULO: **INSTALACIÓN DE UN BRAZO ROBOTIZADO PARA LA AUTOMATIZACIÓN DE LIMPIEZA CON LÁSER DE SUPERFICIES 3D**

AUTOR: **ADRIÁN CORA SIERRA**

TUTOR: **JOSE LUIS CALVO ROLLE**
ALBERTO RAMIL REGO

FECHA: **JUNIO DE 2019**

Fdo.: EL AUTOR

Fdo.: EL TUTOR

Agradecimientos

El presente trabajo es la culminación de cinco años de estudios de grado. Con él se cierra una etapa de mi vida que marcó profundamente la persona que soy a día de hoy.

En tanto a la realización de este trabajo:

Quiero agradecer a mis tutores J. Luís Calvo Rolle y Alberto Ramil Rego, haberme brindado esta oportunidad y su ayuda a la hora de llevarla a cabo.

También quiero agradecer a los profesores y compañeros que me prestaron su ayuda, tanto a nivel técnico, como a la hora de darme sus puntos de vista de un modo más personal.

Tengo que decir que me siento privilegiado al haber podido realizar este trabajo con equipos tecnológicamente punteros[1], ya que me permitieron ampliar mis conocimientos mientras llevaba a la práctica gran parte de los conocimientos que había adquirido cursando el grado.

En tanto a estos últimos cinco años:

Quiero agradecer su atención al profesorado de la E.U.P. que lleva a cabo su labor docente con diligencia todos los días y se preocupa por sus alumnos.

Quiero agradecer a mis compañeros, alumnos de la E.U.P., que hubieran compartido sus apuntes, conocimientos y experiencias de una manera altruista.

También me gustaría agradecer a mis compañeros de piso estos años, en los que me hicieron sentir cómodo y con los que compartí numerosas experiencias comunes.

Por último, quiero agradecer a mi familia y amigos de siempre, que pese a mis numerosas ausencias, siempre estuvieran ahí; ellos son parte elemental de mi. De manera muy especial, se lo agradezco todo a mis padres. Ellos me animaron y apoyaron de manera constante a lo largo de toda esta aventura.

Adrián Cora Sierra
Ferrol, Junio 2019

[1] En este trabajo se han utilizado equipos financiados con cargo al proyecto del Programa Estatal de Investigación, Desarrollo e Innovación orientada a los Retos de la Sociedad del Ministerio de Economía, Industria y Competitividad, referencia BIA2017-85897-R.

TÍTULO: **INSTALACIÓN DE UN BRAZO ROBOTIZADO PARA LA AUTOMATIZACIÓN DE LIMPIEZA CON LÁSER DE SUPERFICIES 3D**

ÍNDICE

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: JUNIO DE 2019

AUTOR: EL ALUMNO

Fdo.: ADRIÁN CORA SIERRA

Listado de figuras

Listado de tablas

Listado de códigos de programación

TÍTULO: **INSTALACIÓN DE UN BRAZO ROBOTIZADO PARA LA AUTOMATIZACIÓN DE LIMPIEZA CON LÁSER DE SUPERFICIES 3D**

MEMORIA

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: JUNIO DE 2019

AUTOR: EL ALUMNO

Fdo.: ADRIÁN CORA SIERRA

Índice del documento MEMORIA

1 OBJETO

Instalación de un robot de 6 ejes para la manipulación (posicionamiento y orientación) de piezas de pequeño tamaño para ser tratadas por láseres pulsados de alta frecuencia. Así como el desarrollo del sistema para coordinar el movimiento del brazo robot, la actuación de los equipos láser y la comunicación de los distintos elementos necesarios para llevar a cabo el proceso.

2 ALCANCE

En el laboratorio de Aplicaciones Industriales del Láser del CIT se dispone de dos láseres pulsados de alta frecuencia con los que se realizan diferentes procesos de micromecanizado por ablación (limpieza, texturizado, etc.) Estos equipos disponen de sistemas de posicionamiento que permiten el desplazamiento del haz láser sobre la pieza a tratar. Para aplicar estos procesos a una pieza no plana se hace necesario el poder orientar la pieza de forma que incida perpendicularmente a su superficie. Para ello se ha realizado la instalación de un pequeño robot industrial que permita la manipulación de las piezas asegurando una velocidad de barrido constante y el mantenimiento de la perpendicularidad entre el haz láser (fijo) y la superficie de la pieza (sujeta por el robot). Este movimiento deberá coordinarse con el inicio/parada de disparo del láser y con el nivel de potencia para lo que se utilizarán las señales de entrada/salida digital/analógica de los diferentes equipos.

Descripción de los hitos previstos del proyecto:

1. Estudio de la documentación técnica de los diversos equipos.
2. Diseño de la distribución de los equipos en el laboratorio.
3. Análisis de las diferentes. posibilidades para el control externo de los láseres.
4. Diseño del cableado y de la secuencia de instrucciones que permiten el control externo de los láseres.
5. Montaje en el laboratorio.
6. Pruebas de desplazamientos y cambios de orientación.
7. Evaluación de los resultados de las pruebas.
8. Generación de la documentación.

3 ANTECEDENTES

3.1. Conceptos básicos de óptica

La luz es la radiación electromagnética que puede manipularse por medio de elementos ópticos como espejos, lentes, obturadores, ... La radiación luminosa corresponde en el espectro electromagnético a las longitudes de onda entre 10 nanometros y 1 micra, incluyendo la región visible y el infrarrojo-ultravioleta cercanos.

La radiación electromagnética (Entre la que se incluye la luz), a nivel macroscópico, se describe mediante unos campos vectoriales que cumplen las leyes de Maxwell, una de cuyas soluciones son las ondas electromagentéticas.

A nivel atómico, la interacción con la materia se realiza mediante partículas o cuantos de luz llamados fotones. El hecho de que la luz se comporte como onda, implica que muchos conceptos que se emplean habitualmente en ingeniería para otro tipo de ondas son aplicables al caso; por ejemplo: las interferencias constructivas y destructivas.

Una particularidad clave de la luz es que esta no necesita un medio físico para propagarse, siendo la velocidad de la luz en el vacío la constante universal $c = 3(10^8)ms^{-1}$.
En la siguiente imagen se muestra la clasificación del espectro electromagnético en función de la longitud de onda (λ) y la frecuencia (f). Véase la distinción entre luz infrarroja, luz visible y ultravioleta entre otras.

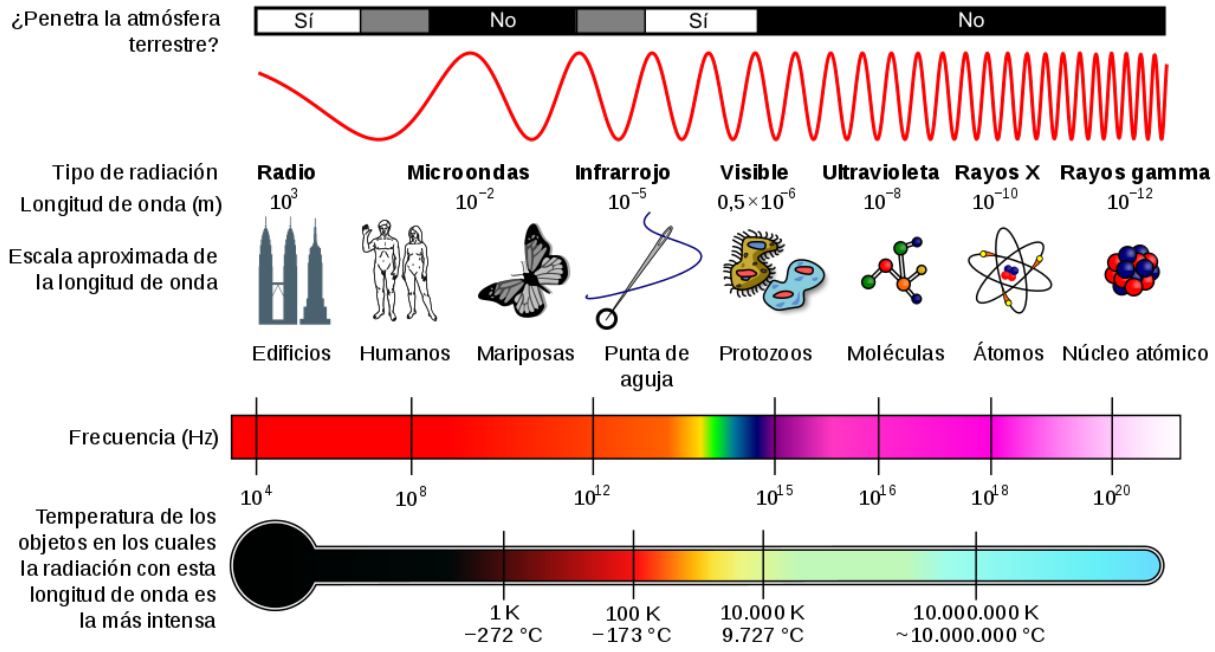


Figura 3.1 – Espectro

Se puede obtener la velocidad de propagación de una onda en función de su longitud de

onda y su frecuencia como:

$$v = \lambda \cdot f \tag{3.1}$$

Cuando se emplea el término de intensidad de un haz de luz habitualmente se hace referencia a la potencia promedio o irradiancia por unidades $[W/m^2]$. En cuanto a la energía de un fotón decir que es producto la frecuencia por la constante de Plank (h).

$$E = h \cdot f \tag{3.2}$$

El efecto fotoeléctrico se puede describir como cuando al iluminar un material con un haz, donde los fotones superan un determinado umbral de frecuencia (y por tanto de energía) los electrones del material se ionizan y comienza su emisión. La absorción más eficiente de energía se da cuando a través de efecto fotoeléctrico, la energía del fotón coincide justo con la cantidad de energía necesaria para subir de nivel el electrón. Cuando estas cantidades no coinciden la interacción entre fotón electrón es mucho menor, atravesando en muchos casos la luz el material sin producirse la interacción. Sin embargo, cabe destacar que la absorción de la luz no tiene porque estar ligada a la emisión de electrones por efecto fotoeléctrico.

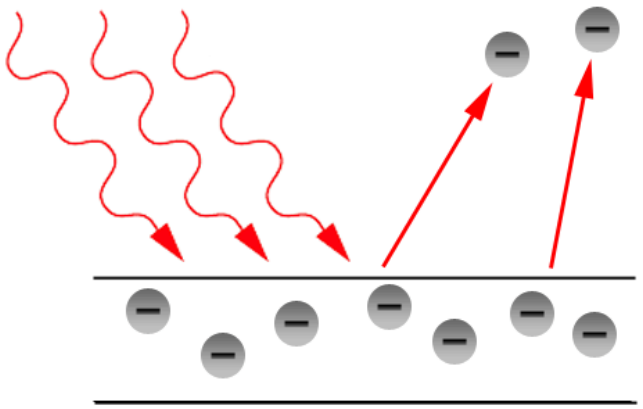


Figura 3.2 – Efecto fotoeléctrico

También es posible el efecto inverso (emisión espontánea), es decir, que un electrón descienda de nivel de energía y emita un fotón. Este fotón portará la energía correspondiente a la pérdida del electrón oscilando a la frecuencia correspondiente.

Existe un tercer efecto relacionado denominado emisión estimulada de luz. Este efecto es el de mayor interés ya que es en el que se basa el funcionamiento de los láseres. En la emisión estimulada, en un campo electromagnético, la presencia de un fotón cuya energía coincida con la diferencia de energía entre dos niveles del material, produce la emisión de un nuevo fotón con las mismas características que el anterior: energía, dirección y polarización.

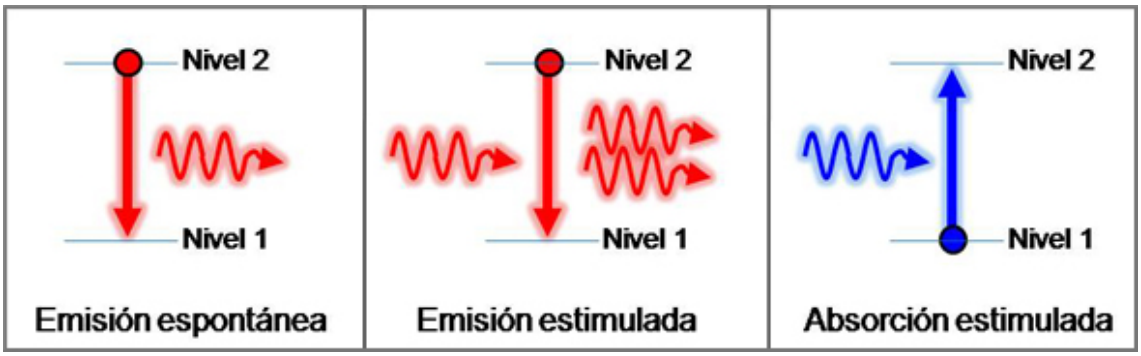


Figura 3.3 – Emisiones de fotones

Véase que a partir de una primera emisión espontánea, en presencia de un campo electromagnético, se produciría un efecto de avalancha, en tal forma que a través de la emisión estimulada se tendría un posterior efecto de amplificación de la luz emitida.

Según la física clásica se define radiación electromagnética (y por tanto también la luz) como dos ondas (una de campo eléctrico y otra de campo magnético). Estas ondas oscilan ortogonalmente entre ellas mismas, y ambas oscilan ortogonalmente con respecto a la dirección de propagación de la luz (la dirección de oscilación de los campos puede cambiar siempre y cuando se cumplan las condiciones citadas). Habitualmente en textos técnicos sólo se cita una onda correspondiente al campo electromagnético. Esto no debe llevar a confusión, se está citando al efecto conjunto de ambos campos de forma abreviada, ya que uno es ortogonal al otro y están en fase.

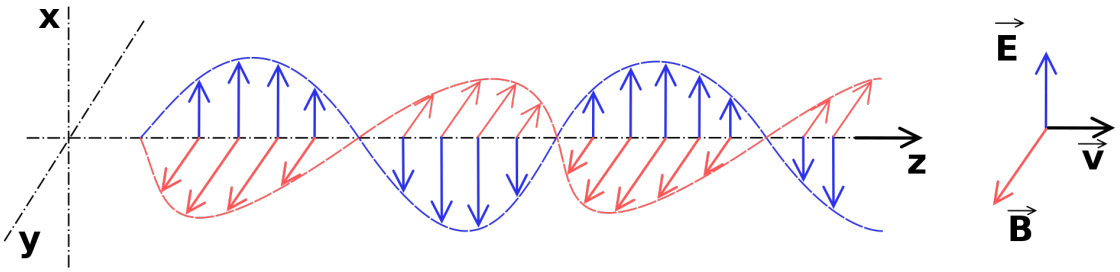


Figura 3.4 – Radiación electromagnética

Decir que la luz está polarizada es lo mismo que decir que la dirección oscilación del campo electromagnético no cambia. La siguiente imagen ilustra un polarizador.

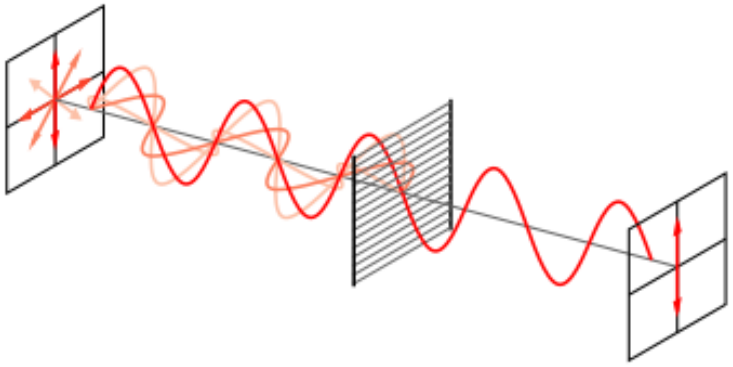


Figura 3.5 – Polarizador

Existen medios en los que la luz no se puede propagar. Estos medios están conformados por materiales denominados opacos.

En los medios en los que la luz sí se puede propagar (no opacos), esta lo hace a distintas velocidades en función de su índice de refracción, el cual se define como el cociente entre la velocidad de la luz y la velocidad de propagación en el medio en cuestión.

En la interfase entre dos medios con distinto índice de refracción se producen los siguientes efectos:

- Refracción: Se conoce como ángulo de refracción al desvío del haz de luz resultado del cambio de velocidad de propagación.
- Reflexión: Es la parte de la radiación que rebota en la interfase y no realiza el cambio de medio.
- Refracción dispersiva: Si la el haz de luz está compuesto de distintas frecuencias cada una de estas se refracta con una dirección diferente.

3.2. ¿Que es un Láser?

Láser (Light Amplification by Stimulated Emission of Radiation). Se entiende por láser un dispositivo que utiliza la emisión estimulada de luz para generar un haz de luz con las siguientes características especiales:

- Monocromaticidad: Los fotones del haz poseen todos la misma longitud de onda (por tanto la misma frecuencia).
- Coherencia: Dicho de otra manera, las partículas emitidas vibran en fase.
- Direccionalidad: La dirección de propagación es única.
- Amplificado: La luz se genera a través de un proceso de amplificación basado en emisión estimulada de luz.

3.3. Funcionamiento de un Láser

En base a los conceptos del efecto fotoeléctrico, emisión espontánea y emisión estimulada, para describir el funcionamiento de un láser conviene manejar también los siguientes conceptos:

- Inversión de población: Se trata de un estado particular de un medio, en el que se posibilita la emisión de luz. Se dice que se da inversión de población cuando existen más átomos de un medio en estado excitado que en estado fundamental.
- Bombeo: Hace referencia al método empleado para aportar energía externa a un medio posibilitándose la inversión de población. Es imprescindible para la emisión de luz. Existen distintos métodos según la tecnología empleada para realizarlo.

La descripción del funcionamiento es en base a un láser típico, no todos estos dispositivos funcionan exactamente igual y tienen las mismas partes pero de forma generalista. La siguiente imagen ilustra el funcionamiento de un láser:

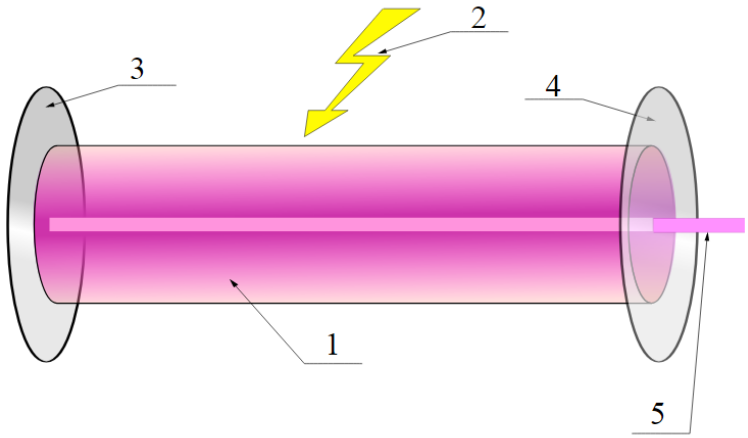


Figura 3.6 – Cavidad láser

En la imagen se indican las siguientes partes:

1. Medio activo: Es el material con las propiedades adecuadas para que se produzca la emisión estimulada de luz.
2. Energía bombeada: Hace referencia simbólica al método tecnológico empleado para realizar el bombeo de energía.
3. Espejo de alta reflectancia: Junto al otro espejo fija la direccionalidad de la luz. Refleja la práctica totalidad de la luz que le impacta en sentido opuesto.
4. Espejo de reflactancia parcial: Refleja parte de la luz y deja escapar parte de esta emitiendo el haz.
5. Emisión del haz láser: La luz emitida tiene las propiedades propias de la luz láser (coherencia, direccionalidad. . .).

3.4. Tipos de Láser

Existe una infinidad de láseres desarrollados con una gran diversidad de tecnologías. Una de las maneras más sencillas de diferenciarlos es según el tipo de medio activo empleado. Según su medio activo distinguimos:

- Láseres de gas: Utilizan como medio activo un gas. El tamaño del medio activo no es una restricción. El medio activo puede ser excitado de muchas formas distintas (descarga eléctrica al gas, inyectando haz electrones o iones, radiación electromagnética de baja frecuencia. . .). Algunos de los gases más habituales son: Helio-Neón, CO2, Argón. . .

- Láseres de estado sólido: El medio activo es un cristal o un vidrio dopados con una pequeña cantidad de iones (1 %). Se destacan por su facilidad de fabricación. Son habituales los medios activos: Rubí, Iones de neodimio.

- Láseres de semiconductor: El medio activo es un semiconductor una unión de semiconductores. Son los más vendidos porque en comparación con los otros tipos su tamaño es mucho más reducido y su fabricación más sencilla. Su funcionamiento se basa en la aplicación de un campo eléctrico a un semiconductor, cuando este comienza a conducir, en ciertos semiconductores se produce la emisión de fotones. Si la estimulación es lo suficientemente alta, comienza la emisión coherente de luz. Si la inversión de población no es lo suficientemente alta el semiconductor se comporta como un LED. Cabe destacar que la dirección de emisión de la luz coherente es paralela a la unión del semiconductor y que el haz tiende a tener forma de elipse.

- Otros tipos: Existen otras tecnologías para generar emisión láser aunque estas tienen una menor popularidad.

3.4.1. Láseres pulsados

Típicamente se define un láser como un haz de luz con las propiedades intrínsecas de mocromaticidad, coherencia y direccionalidad. Por tanto es habitual dar por hecho que se trata de un haz continuo de luz a lo largo del tiempo. Esto no tiene por qué ser así, ya que también es muy habitual que la emisión del haz se produzca en forma pulsada, es decir, que se trate de un láser pulsado y no de emisión continua.

Hay que distinguir entre la frecuencia de la luz (relacionada directamente con la longitud de onda) y la frecuencia de pulsado del haz. Se ilustra en la siguiente gráfica, donde la primera es la frecuencia de la portadora, mientras que la segunda la de la envolvente:

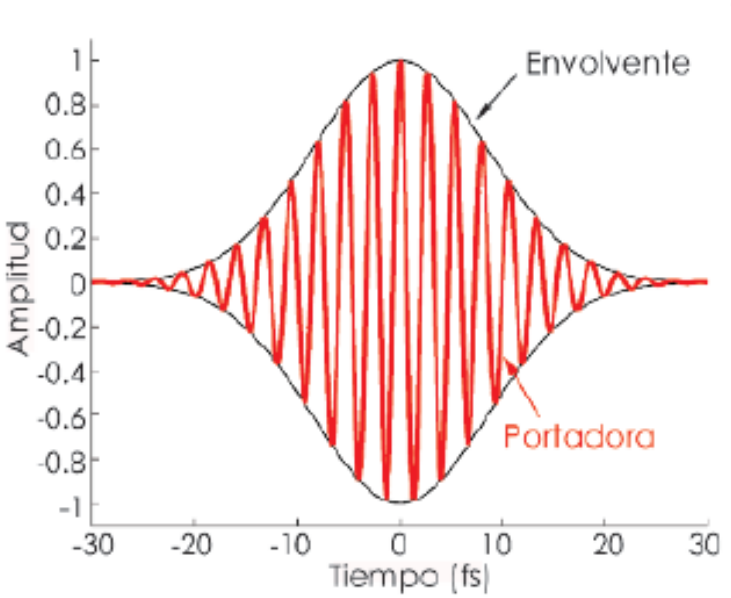


Figura 3.7 – Envolvente

Existen dos técnicas principalmente para lograr la emisión pulsada en un láser:

- Q-Switching: Consiste en bloquear la emisión del láser, evitando que el haz salga de la cavidad del láser. De esta manera se produce la inversión de población concentrado energía en el medio. Una vez que se supera el umbral de energía deseado en el medio activo se desbloquea la salida del haz permitiendo la emisión. Esto se puede conseguir con técnicas activas o pasivas. La principal diferencia es que empleando técnicas activas se controla a voluntad cuando se permite la emisión, mientras que con técnicas pasivas se emplea un material especial en la salida que permite la emisión solo cuando la inversión de población supera cierto umbral.

- Mode Locking: Este método se basa en tener ondas de luz en el medio en distintos modos bloqueados. En otras palabras, tener distintos armónicos cuya fase esté bloqueda, es decir que sea constante. Cuando todos los armónicos estén en fase se producirá una interferencia constructiva y la emisión será máxima produciéndose un pulso. Pese a ser una técnica más compleja, de manera análoga existen técnicas activas y pasivas para realizar el control.

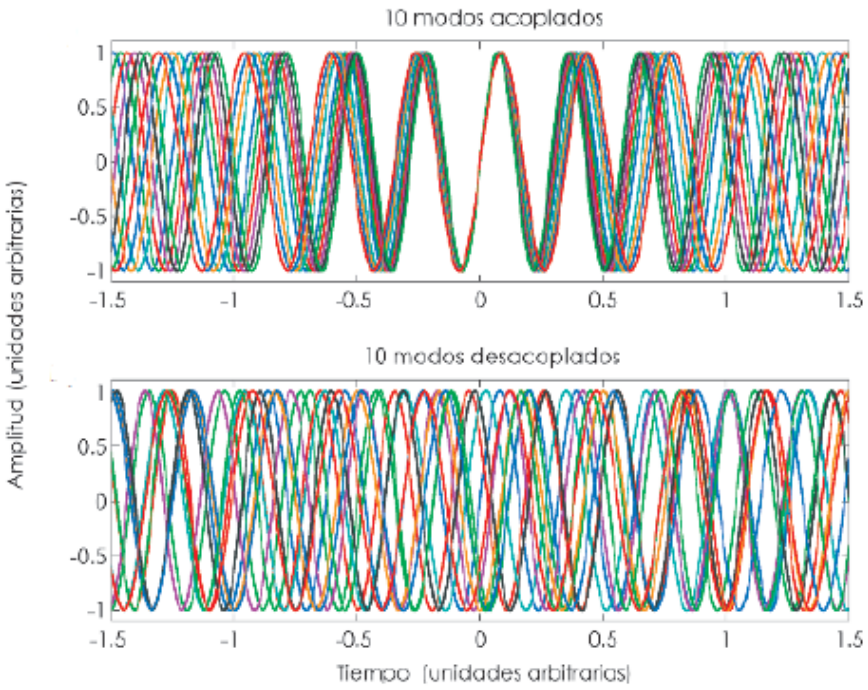


Figura 3.8 – Mode Locking

3.5. Aplicaciones Industriales del Láser

A día de hoy el láser es ampliamente empleado en la industria. Algunas de sus principales aplicaciones son:

- Corte: Se emplean láseres de emisión continua y de alta potencia. Sus principales ventajas son la flexibilidad del proceso así como el muy bajo o nulo desgaste de la herramienta. El corte puede ser muy estrecho y preciso y las distorsiones térmicas suelen ser relativamente bajas (en muchos casos microscópicas).
- Soldadura: El proceso es similar al corte, la diferencia es que normalmente se aporta un gas que actúa como atmósfera protectora. Conforme avanza el haz a lo largo de la superficie se va creando el cordón de soldadura por fusión de los materiales.
- Soldadura híbrida láser-arco: Combinación de los procesos de soldadura láser y soldadura GMAW (Gas Metal Arc Welding). Se trata de un proceso de unión de materiales. El proceso se basa en el desplazamiento de un arco que funde el metal aportado dentro de la atmósfera mientras que el láser calienta la superficie de unión favoreciendo el aporte y consiguiéndose de este modo elevadas velocidades de soldadura.
- Taladrado: Se consigue propiciando con un láser la evaporación del material. El diámetro del taladro es proporcional a la longitud de onda del haz. Tiene limitaciones de profundidad.
- Cladding: Es una técnica de fabricación aditiva. Consiste en el aporte de polvo que mediante un láser no pulsado se funde y se deposita sobre la superficie del material a tratar.

Además del propio láser y del sistema de posicionamiento y translación es necesario una serie de sensores (Pirómetro, fotodiodo. . .) para regular y/o monitorizar el proceso.

- Grabado: Consiste en el grabado de información empleando un láser que trabaja de forma superficial.
- Tratamientos superficiales: Conjunto de técnicas que tienen por objetivo modificar características y propiedades de la superficie de un material. Son tratamientos superficiales:
  - Temple Superficial.
  - Recubrimiento: En la superficie mediante aporte de material.
  - Incremento de la rugosidad.
- Procesado con láseres pulsados de alta frecuencia: Permite el procesado de materiales transparentes o de muy alta dureza. Al enfocar un láser de pulsos de femtosegundos sobre un material (dieléctrico, metal o semiconductor) se produce ablación ultrarrápida. A grandes rasgos la ablación consiste en aplicar un pulso de una intensidad elevada sobre un material durante un intervalo de tiempo del orden de magnitud de femtosegundo. Con esto se consigue romper los enlaces entre partículas mediante la aportación de energía pero sin efectos térmicos dada la corta duración del pulso, es decir, se trata de un mecanizado. La ablación ultrarrápida también suele denominarse micromecanizado. El micromecanizado y abarca el conjunto de procesos donde se extrae el material mediante un láser pulsado por fusión, vaporización o ablación del mismo.

3.5.1. Integración de sistemas

Generalmente todos los sistemas industriales que emplean el láser se pueden descomponer en la siguiente serie de elementos o subsistemas:

- Láser: Las características que debe tener el equipo a usar dependen de forma directa de los requerimientos del proceso.
- Sistema de translación: Existen varias tipologías posibles de sistemas de translación. Normalmente se requiere que el haz láser trabaje normal a la superficie de trabajo, por este motivo es habitual el uso de sistemas translación con más de tres GDL (Grados de libertad) que lo permitan. En otros casos el empleo de sistemas de translación cartesianos u otros de pocos grados de libertad es suficiente.
- Sensores: Algunos procesos industriales basados en el láser requieren de sensores, para que a través de los cuales se regule el proceso empleando técnicas de ingeniería control. En otros casos esto no es un requerimiento estricto, pero empleando este tipo de controles habitualmente se puede afinar el desempeño del proceso en gran medida, o simplemente puede que sea interesante llevar a cabo una monitorización del proceso. En cuanto los tipos de sensores empleados, son muy diversos. Estos van desde sensores analógicos que emplean como interface con el resto de sistema de control tarjetas

de adquisición DAQ (Data Acquisition), o módulos de entrada salida digitales, hasta sensorizaciones mucho más complejas como cámaras de alta velocidad o escáneres de superficie.

- Comunicaciones: Este tipo de procesos son complejos en cuanto sus requerimientos y a la cantidad de elementos heterogéneos. Habitualmente es necesaria la integración de distintos subsistemas y elementos que emplean estándares de comunicación distintos. En cierta manera esto está justificado por la cantidad de información, velocidad, complejidad y sincronismo de la misma que es necesario transmitir en cada punto. Pero lo cierto es que en muchos casos el hecho del empleo de elementos no específicos de procesos industriales con láser, y que los fabricantes compliquen deliberadamente la integración de sus productos en sistemas mixtos, puede llegar a ser una problemática a la hora de desarrollar una solución en cuanto a comunicaciones. Por este motivo es habitual el uso de pasarelas de comunicaciones que actúen como interface entre dos sistemas que emplean estándares de comunicación a priori no compatibles.

- Sistemas de Control: Están presentes en todo el proceso, desde el sistema de translación hasta en la regulación de los parámetros del haz. Un factor determinante en la elección y desarrollo de los elementos o subsistemas que conforman el sistema es el tiempo de respuesta, entendiendo tiempo de respuesta como el retardo entre la generación de una señal de control y la estabilización de su valor dentro de los márgenes deseados. Para dimensionar de forma acertada el sistema y sus componentes se debe asegurar que el tiempo de respuesta del sistema de control es inferior a la dinámica del sistema a controlar. De manera análoga si el proceso está sensorizado, el muestreo y estado de los sensores debe evolucionar más rápidamente que la dinámica del sistema.

3.5.2. Integración de sistemas

Se conoce como TCP (Tool center point) el punto de la herramienta (láser) a través del cual se describe la posición de la misma respecto a un sistema de referencia normalmente absoluto y estático. Asimismo, este punto suele ser origen de coordenadas de un sistema de referencia asociado a la herramienta que aporta información acerca de su orientación.

En el diseño conjunto de un sistema industrial para procesamiento con láser es importante tener en cuenta las restricciones de trayectorias que tendrá que describir la herramienta. Estas restricciones son habitualmente más estrictas que en otras máquinas CNC (Control Numérico Computarizado) debido a las características propias de las herramientas láser a la hora de procesar.

Algunas de las restricciones más habituales son:

- Velocidad constante: La velocidad de desplazamiento del TCP deberá ser constante. Esto se justifica porque el grado de procesado de un punto dependerá del tiempo de exposición al haz, y lo que es deseable habitualmente es una exposición igual en todos los puntos de la trayectoria.

- Distancia focal constante: Por definición un láser es un haz direccional. En principio, según esta definición, la distancia de una herramienta láser a una pieza no es importante. Sin embargo se suelen emplear lentes para focalizar el haz de luz y concentrar así el haz. De esta manera se consigue una densidad de potencia por área más elevada a costa de perder la direccionalidad, y en consecuencia tener un punto de máxima focalización. Por otro lado, en el caso de que el proceso esté sensorizado una distancia no constante puede llevar a errores de medición.

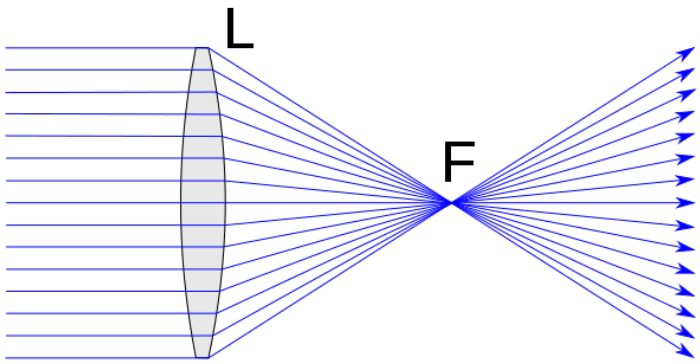


Figura 3.9 – Lente convexa

- Orientación correcta: Es importante en cuanto a la eficiencia de la herramienta y producción de reflejos. Lo deseable suele ser una orientación normal del eje de la herramienta con respecto a la superficie a tratar.

- Rotación limitada: En muchos casos la rotación de la herramienta en torno a su propio eje está limitada. Aunque el sistema permita esta rotación respecto al análisis de sus GDL existen limitaciones prácticas. Estas limitaciones suelen ser conexiones cableadas de sensores o de la propia herramienta.

- Aceleración no infinita: Las trayectorias que describe el movimiento del TCP durante el proceso han de ser derivables, dicho de otra manera, estas han de ser continuas y no han de presentar picos. Los picos equivales a cambios de velocidad instantáneos, por tanto aceleraciones infinitas, las cuales no se pueden dar en sistemas de translación reales. Las aceleraciones necesarias para describir las trayectorias no han de superar las aceleraciones máximas que se pueden conseguir con el sistema de translación.

3.6. Brazo robot

Según AFNOR (Association française de Normalisation), un manipulador es un Mecanismo formado generalmente por elementos en serie, articulados entre sí, destinado al agarre y desplazamiento de objetos. Es multifuncional y puede ser gobernado por un operador humano o mediante un dispositivo lógico.

Según las normas ISO, un robot industrial es un manipulador multifuncional con varios grados de libertad, capaz de manipular cargas, piezas, herramientas o dispositivos especiales, según trayectorias programadas para realizar tareas diversas.

Los siguientes conceptos son empleados habitualmente en la descripción de este tipo de máquinas:

- Antropomórfico: Se dice que un brazo robot es antropomórfico cuando la forma y distribución de sus articulaciones tiende a parecerse a las de un brazo humano (habitualmente con seis GDL).
- Cartesiano: Un sistema de translación cartesiano es aquel que tiene tres GDL formados por desplazamientos lineales correspondientes cada uno de ellos a la dirección de un eje de un sistema de referencia cartesiano.
- GDL (Grados de libertad): Cada uno de los movimientos independientes que puede realizar cada articulación con respecto a la anterior. Los GDL de un robot es la suma del de sus articulaciones.
- Volumen de trabajo: Es el volumen espacial al que puede llegar el elemento terminal del sistema de translación. También se suele emplear el término área de trabajo cuya definición es análoga.

3.7. Comunicaciones

En la elaboración de este trabajo se emplearon los siguientes tipos de comunicaciones:

3.7.1. Comunicación serie

Comunicación serie es aquella que utiliza un medio o bus para trasmitir información y lo hace de forma secuencial, es decir, bit a bit. En contraposición, la comunicación en paralelo transmite un conjunto de bits en paralelo (a la vez) y no tiene por qué ser estrictamente síncrona.

Entre las ventajas de la comunicación serie frente la comunicación paralelo está que se necesitan menos líneas o canales de comunicación. Esto sin embargo, implica que para enviar la misma cantidad de información es necesario elevar la frecuencia de transmisión. Otra de las ventajas asociadas con la comunicación serie es la reducción de costes.

A la hora de establecer una comunicación serie entre elementos existen distintos estándares normalizados con distintas características. El uso de un estándar u otro dependerá de los requerimientos de la aplicación. A continuación se describen los estándares de comunicación serie empleados en este trabajo junto algunas de sus características.

3.7.1.1. RS-232

Es el estándar que más se ajusta a la forma de comunicar que emplean los PC en su puerto serie (acostumbran a incorporar conectores DB-9). La típica codificación empleada en

este protocolo de comunicación es el estándar ASCI que emplea 7 bits (8 en el caso del ASCI extendido) para codificar la mayor parte de caracteres del alfabeto latino, y algunos otros de control. En el protocolo de comunicación es habitual el envío de un primer bit de start y un bit de stop para cerrar la comunicación. De estos parámetros dependerá el tamaño de dato y del mensaje. Emisor y receptor deben funcionar de igual manera para una comunicación correcta.

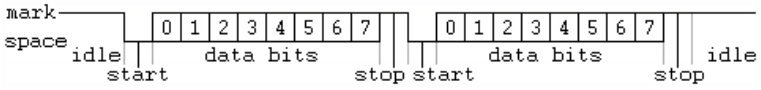


Figura 3.10 – Trama RS-232

En cuanto al protocolo, la codificación ASCI se realiza sobre un canal empleando niveles lógicos de tensión que de forma secuencial codifican los datos bit a bit. Otras características a tener en cuenta son:

- Niveles lógicos: En el standard RS-232 los niveles lógicos vienen determinados por los voltajes entre los distintos pines en cada conector respecto uno de ellos es hace la función de masa común. El protocolo empleado en el estándar RS232 se dice que es NRZ (non-return-to-zero) porque no hay un nivel lógico neutro. Este estándar comprende dos niveles lógicos y ambos son significantes.

Voltaje	**Nivel lógico '1'**	**Nivel lógico '0'**
Min	$+3V$	$-15V$
Max	$+15$	$-3V$

Tabla 3.1 – Niveles lógicos RS-232

- Tasa de bits: La máxima tasa transmisión de bits es de 19.200 bps, es decir 19200 bits/s. Esta tasa es configurable en muchos sistemas y aparece expresada como un baudrate; por ejemplo de 9600 o de 115200. Esto es lo mismo expresado en baudios que en el caso de RS-232 equivale a bits/s, pero no siempre tiene que ser así, ya que el baudio expresa más bien el tamaño de paquete que en este caso por convenio equivale al bit.

- Máxima longitud: la máxima longitud del cable entre puntos empleando la comunicación RS-232 es de 15m. Este dato depende mucho del nivel de ruido del entorno, y del correcto trenzado y blindaje de los conductores.

- Topología: Este tipo de comunicación se define como full-duplex y punto a punto. Full dúplex porque es un tipo de comunicación donde emisor y receptor pueden comunicar a la vez, y leer mientras escriben. Punto a punto porque no se trata de un bus, sino que es una conexión entre dos elementos.

- Conexión física: La conexión física más habitualmente empleada es el conector DB-9, ya que muchos disponen de este puerto serie.

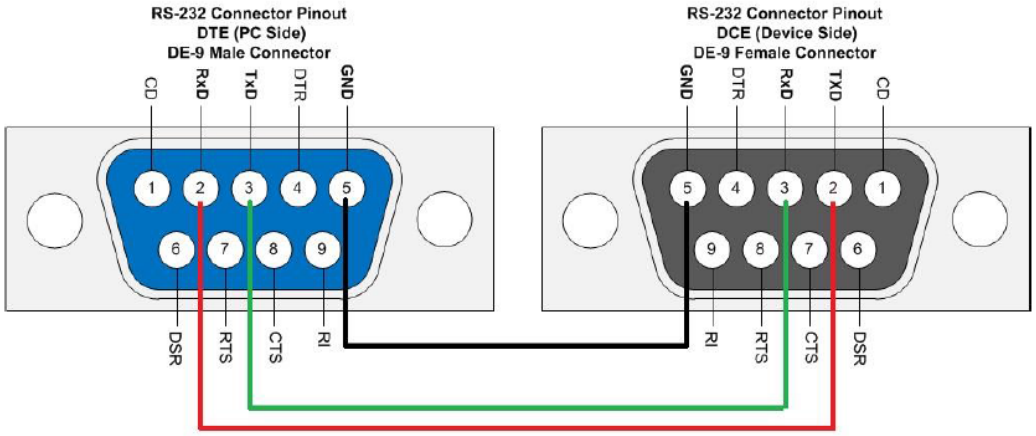


Figura 3.11 – Conector DB-9 RS-232

El pinout para el estándar RS-232 es igual para todos los conectores, disponiendo eso sí, de algunos pines adicionales para funciones muy específicas que no tienen gran interés, en este trabajo y cambiando la numeración de los pines. En la siguiente tabla se relaciona cada pin de un conector DB-9 macho con su nombre y función:

Pin	Nombre	Función
1	CD	Entrada, portadora detectada (Carrier Detect)
2	RXD	Entrada, datos recibidos (Received Data)
3	TXD	Salida, datos transmitidos (Transmitted Data)
4	DTR	Entrada, terminal de datos o modem listo (Data Terminal Ready)
5	GND	Masa común referencia (Common Ground)
6	DSR	Entrada, equipo de datos listo (Data Set Ready)
7	RTS	Salida solicitud de envío(Request To Send)
8	CTS	Entrada, listo para enviar (Clear To Send)
9	RI	Entrada, Indicador de Anillo (Ring Indicator)

Tabla 3.2 – Pines concector DB-9 RS-232

3.7.1.2. TTL (UART/USART)

La mayor parte de las características de este tipo de comunicación serie son similares al RS-232. El motivo de las diferencias es que este tipo de comunicación está pensada más bien para conectar equipos como microcontroladores y sus periféricos, por lo que suelen ser conexiones internas en un equipo en lugar de entre equipos. Cabe destacar que TTL permite una distancia como máximo de 2m en los conductores sin problemas de atenuación y ruido.

- TTL (Transistor-transistor logic): Es una tecnología de fabricación de circuitos electrónicos digitales con transistores. Los rangos habituales de la tensión de alimentación y de los niveles lógicos se resumen en la siguiente tabla:

Nivel	Alimentación	Nivel lógico '1'	Nivel lógico '0'
Max	$Vcc >= 4,75V$	0	$2,2V$
Min	$Vcc <= 5,25V$	$0,8V$	Vcc

Tabla 3.3 – Niveles lógicos TTL

No está de más mencionar que es habitual diferenciar los rangos de niveles lógicos entre entrada y salida siendo denominados como: V_{IL} Y V_{IH} los niveles mínimos y máximos para la entrada respectivamente; V_{OL} Y V_{OH} los niveles de forma análoga para la salida.

- UART (Universal Asyncronous Receiver/Transmiter): Es un dispositivo que se encarga de controlar los dispositivos serie que se encuentran conectados a la placa base o tarjeta del equipo. Desde el punto de vista de un microcontrolador es un periférico (suele venir en el circuito integrado del microcontrolador) cuya función es transformar la información con la que trabaja la CPU, típicamente palabras de 8 bits, y enviarlo de forma secuencial en serie con el formato y protocolo adecuado. También realiza la función inversa, adaptando la información entrante al formato en que trabaja la CPU y la almacena en un buffer hasta que sea demandada. Las velocidades de transmisión pueden llegar a ser de hasta 921,6Kbps.

- USART (Universal Syncrchronous/Asynchronous Receiver/Transmitter): Es un dispositivo muy similar a la UART, de hecho en muchos casos se hace trabajar como UART. La diferencia principal es que permite una comunicación serie síncrona además de asíncrona. Además, la denominación USART suele emplearse para dispositivos que aunque trabajen como UART, lo hacen con mayores prestaciones; en algunos casos con tasas de transmisión de más de 4Mbps.

3.7.1.3. Puertos E/S (Entrada/Salida)

Con el objetivo de poder ampliar las funcionalidades o de proporcionar mayor flexibilidad, los equipo industriales con control electrónico habitualmente incorporan una serie de puertos con conexiones denominadas entradas y salidas en función de si transmiten o reciben información. En función del tipo de información con el que trabajen las entradas y salidas, pueden ser de dos tipos:

- Digitales: La información es de tipo discreto, trabajan con niveles lógicos según los cuales una magnitud física podrá tomar únicamente dos valores fácilmente diferenciables en función del nivel. Los valores de las magnitudes físicas normalmente cumplen con un estándar, el estándar viene determinado por las necesidades y restricciones de la aplicación. Es habitual el uso de optoacopladores para comunicar equipos a través de estos puertos y, algunos equipos incluso los incorporan internamente. Estos dispositivos permiten el intercambio de información digital aportando aislamiento galvánico entre las partes, lo cual permite una adecuación sencilla de los niveles lógicos en muchos casos

y la protección ante fallas en uno de los equipos. Existen distintos tipos de optoacopladores según los componentes que los integran: fotoransistor, fototriac, fototiristor... Pero en todos ellos el principio de funcionamiento es el mismo, Cuando se alimenta un LED en su entrada este excita un componente fotosensible en el interior y hace que su salida conmute.

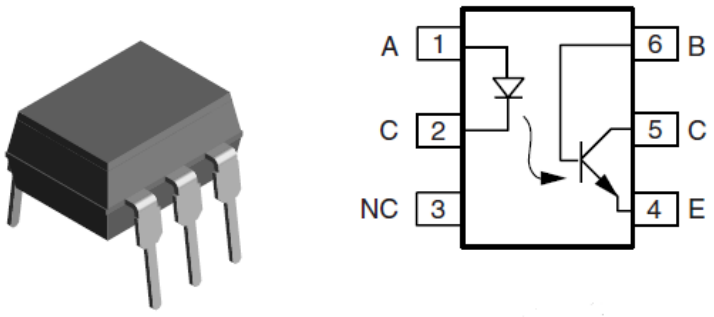


Figura 3.12 – Optoacoplador fototransistor

- Analógicas: La información es un valor de una magnitud continua (habitualmente voltaje), lo que quiere decir que teóricamente un valor analógico proporcional a una magnitud física puede tomar infinitos valores dentro de un determinado rango. Esta definición es cierta, pero en la práctica está un poco más acotada por la necesaria aproximación que se produce cuando se emplean sistemas reales. Las restricciones más representativas del caso real son:
  - Resolución: Cuando estas señales se generan o miden con equipos lógicos como puede ser un microcontrolador para poder expresarse en forma de lógica programable se han de cuantificar. Según la resolución con la que se haga será una medida más precisa y habitualmente más costosa de realizar. La resolución está directamente relacionada con el número de bits empleados para expresar el valor numérico de la medición, a mayor número de bits, mayor resolución. El número de valores numéricos para expresar una señal es igual a dos elevado al número de bits.
  - Rango o fondo de escala: El rango en la magnitud física de una señal analógica es la diferencia entre su valor máximo y mínimo.
  - Exactitud: Es el grado de concordancia entre el valor exacto de la entrada y el valor medio, se expresa normalmente en % sobre el fondo de escala.
  - Precisión: Es el grado de concordancia entre los resultados en medidas sucesivas.
  - Repetitividad: Está directamente relacionada con la precisión y la exactitud.

3.7.1.4. Otras tecnologías empleadas

- WiFi: Tecnología que permite la interconexión inalámbrica de varios dispositivos electrónicos de forma simultánea. Está definida en el estándar IEEE 802.11. Acostumbra a em-

plear protocolos como TCP/IP, UDP...

- Ethernet: Conexión de acceso múltiple con escucha de portadora y detección de colisiones. Es un estándar para la comunicación entre equipos electrónicos. Acostumbra a emplear protocolos como TCP/IP, UDP...

- TCP (Transmission Control Protocol): Protocolo para la comunicación de redes de ordenadores, perteneciente a la familia de protocolos de internet. Se caracteriza por asegurar la transmisión segura de datos, o lo que es lo mismo, asegurar a través del reenvío de información desde a receptor a emisor de que la transmisión de datos fue correcta.

- UDP: Protocolo para la comunicación de redes de ordenadores, perteneciente a la familia de protocolos de internet. Se caracteriza por no asegurar la transmisión segura de datos, es decir, no se comprueba que los datos se enviaron y recibieron correctamente.

- Protocolos TCP/IP: Así se denomina de forma genérica al conjunto de protocolos de red en los que se basa internet y muchas otras redes de estructura análoga que permiten el intercambio de datos entre computadores.

- DeviceNet: Bus de campo basado en el CAN Bus con la particularidad y diferencia respecto a este último de ser orientado al envío de objetos de comunicación. Emplea el protocolo CIP (Common Industrial Protocol). Está diseñado en tal forma que cada componente real se representa en el protocolo como una representación abstracta en forma de objeto, la estructura lógica es muy similar a la de los lenguajes de programación orientados a objetos. Se trata de un protocolo basado hardware y software abierto originalmente desarrollado por la compañía Allen-Bradley (actualmente perteneciente a Rockwell Automation).

- Pasarela de comunicaciones: Denominada en muchos casos gateway, se trata de un elemento de comunicaciones que trabaja con dos o más estándares o protocolos. Su función es la de hacer de puente entre elementos que emplean distintas formas de comunicación actuando como si se tratara de un traductor.

3.7.1.5. Microcontrolador

Abreviado como μC, UC o MCU es un circuito integrado programable compuesto por una CPU, una memoria RAM típicamente volátil, una memoria de programa ROM (Últimamente Flash) y una serie de periféricos auxiliares. Se trata de un circuito secuencial síncrono cuyas frecuencias son características de las que depende su ciclo máquina y suelen ser del orden de varios MHz. Son componentes de coste relativamente bajo y habitualmente de propósito general aunque en función de sus características serán más o menos adecuados para una aplicación u otra.

Figura 3.13 – Microcontrolador ATMEGA328P

Una de las principales diferencias de un microcontrolador respecto a un microprocesador (aparte de que suele integrar en el mismo encapsulado más elementos) son sus periféricos integrados. Estos periféricos son circuitos lógicos normalmente digitales y síncronos cuyas funciones son muy específicas. Presentan la ventaja de que liberan a la CPU de realizar tareas que ralentecerían el sistema en gran medida, de este modo, se pueden conseguir grandes prestaciones con una CPU relativamente sencilla. Los siguientes elementos son periféricos habitualmente integrados en un microcontrolador:

- Convertidor AD (Analógico Digital): Su función es la de convertir valores de señales analógicas a valores digitales discretos cuantificándolas. Normalmente las magnitudes con las que trabaja este periférico son tensiones entre 0 y 5V. Al emplear uno de estos convertidores siempre hay una pérdida de información por el hecho de cuantificar valores continuos. Sin embargo, si la señal está bien acondicionada y el convertidor es el adecuado la pérdida de información es despreciable o no significativa. Cuando se mide la evolución una señal a lo largo del tiempo también hay una pérdida de información por el hecho de que no se adquiere la medida de forma continua, sino que se muestrea, normalmente, cada cierto intervalo de tiempo constante (periodo de muestreo). Para realizar medidas correctas de una señal se ha de asegurar que el periodo de muestreo es lo suficientemente pequeño para poder adquirir la información dinámica deseada de la señal medida. Existen diversas tecnologías de convertidores AD, más o menos adecuadas según el propósito del mismo, pero en general a mayores prestaciones, mayor coste y a mayor rapidez de conversión, menor resolución.
- Convertidor DA (Digital Analógico): Su funcionamiento es inverso al del convertidor AD, consiste en reconstruir valores analógicos de señal a partir de valores discretos. En los micrcontroladores es habitual que el valor analógico reconstruido se exprese como un voltaje entre 0 y 5V. Idealmente el periférico expresaría el valor como una tensión fija cuando se le asigna un valor discreto a reconstruir, pero en la práctica muchas veces esto no es así. La razón es que es más fácil de integrar y definitivamente menos costoso. Dado que en vez de emplear en su constitución un circuito digital-analógico que exprese

un valor fijo se emplee un circuito digital que genere una señal PWM cuya tensión media sea equivalente a ese valor fijo. A la hora de generar esta señal no se reducen las prestaciones de procesamiento del micro controlador, porque se suele hacer a través de un periférico denominado timer con distintas funcionalidades (entre ellas esta). Un timer es un periférico cuya función es la de realizar una temporización. Esta temporización se puede hacer de distintas maneras, por ejemplo: La CPU puede ordenar el inicio de la temporización y consultar cuando lo necesite el tiempo transcurrido, antes eso sí del overflow o desborde de la cuenta. También se puede configurar el timmer que cuando trascurra un tiempo desde el inicio de la cuenta genere una interrupción a modo de aviso a la CPU. En el caso de la generación de una señal PWM el timer conmuta una salida física del microcontrolador (normalmente entre 0 y 5V) de forma periódica. El valor medio de tensión en la señal es configurable y depende del punto de la temporización en que se produzca la conmutación (el ciclo de trabajo). Por tanto una señal PWM es una señal cuadrada de periodo constante cuyo punto de conmutación temporal es variable en función de su ciclo de trabajo (dutty cycle) y de ello depende su valor medio.

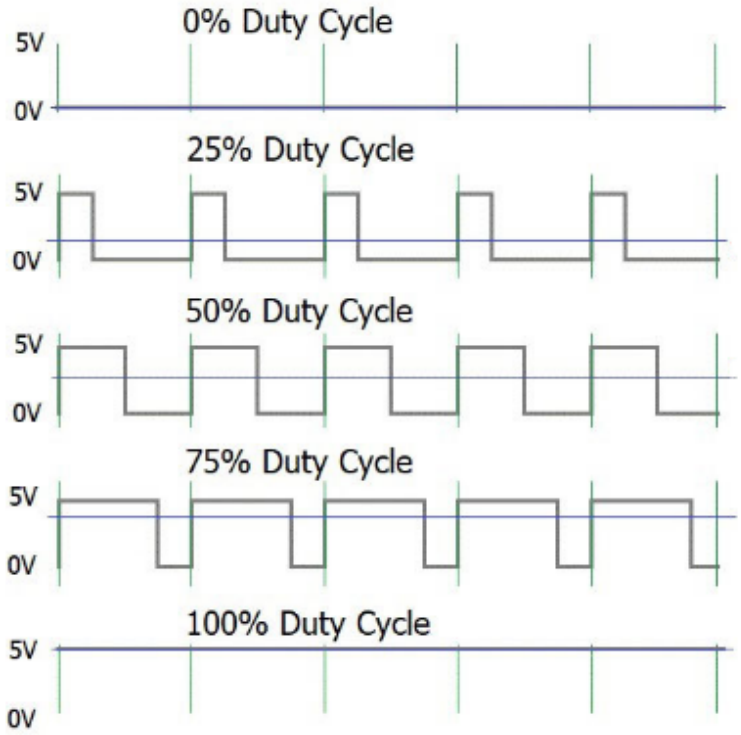


Figura 3.14 – PWM con distintos ciclos de trabajo

- Puertos IO: Los puertos de entrada salida de un microcontrolador están formados por biestables que conmutan entre dos estados (normalmente 0 y 5V) un pin del encapsulado respecto al pin de masa. El estado de estos puertos puede ser modificado por la CPU, aunque en muchas ocasiones están compartidos con los periféricos para dar salida independiente de la CPU a las funcionalidades de estos.

- UART/USART: Normalmente la UART o USART (el puerto serie) se entiende como un

periférico desde el punto de vista del microcontrolador.

3.7.1.6. DAQ (Data Acquisition)

DAQ o adquisición de datos es el proceso de medir con un PC u otro dispositivo lógico una magnitud física como voltaje, corriente, temperatura o presión. Un sistema DAQ consiste de sensores, hardware de medida DAQ y un dispositivo como un PC del cual se aprovecha la capacidad de procesamiento para procesar los datos adquiridos. Es una solución para realizar medidas potente, flexible y rentable.

Una tarjeta DAQ, también denominada tarjeta de adquisición de datos es un dispositivo que incorpora el hardware necesario para servir de interface entre un dispositivo lógico y señales analógicas. Por definición consta de un convertidor AD de varios canales con capacidad para medir distintas señales analógicas que suelen ser voltajes. Algunas tarjetas de adquisición de forma adicional disponen de canales de salida que pueden a través de los cuales es posible reconstruir señales o valores estáticos, suelen habitualmente voltajes de forma análoga a las entradas. Un factor determinante en estos dispositivos es su velocidad de adquisición de datos, o dicho de forma más correcta la máxima frecuencia de muestreo a la que pueden medir o reconstruir valores. Este factor junto a lo fidedignas que sean las medidas y reconstrucciones conforman sus prestaciones más importantes. En función de estas prestaciones y otras como puede ser la fiabilidad, filtrado incorporado, inmunidad frente a ruidos... un equipo será más o menos económico.

3.8. Estado inicial de los equipos

En este apartado se explica el estado y configuración iniciales de los equipos y los métodos empleados de comunicación implementados en el laboratorio de aplicaciones industriales del láser.

En el comienzo este trabajo se había instalado un pequeño brazo robot manipulador de ABB sobre una mesa móvil junto a su controladora en el laboratorio. La controladora del brazo tiene entre otros elementos una computadora con conectividad ethernet, por lo que el punto de partida se considerará como los equipos anteriormente instalados para realizar otros procesos de micromecanizado en laboratorio junto al brazo y su controladora sobre la plataforma móvil conectado a la red local mediante cable ethernet.

3.8.1. Listado de equipos del laboratorio

- Láser New Port, Spectra Phisics, Spirit 1040-4
- Láser Coherent, AVIA Ultra 355-20008
- Brazo Robot ABB, IRB 120 integrado junto su controladora ABB, IRC5 Compact
- Equipos informáticos y otros sistemas de translación:

- Dos Controladores Cartesianos con GDL correspondientes a los ejes XY y XYZ respectivamente.
- Cámaras y otros sensores periféricos cuya comunicación es RS-232
- Switch Ethernet
- PC del laboratorio

3.8.2. Diagrama de red de comunicaciones inicial del laboratorio

En el siguiente diagrama se muestran las comunicaciones implementadas entre los equipos en el punto de partida:

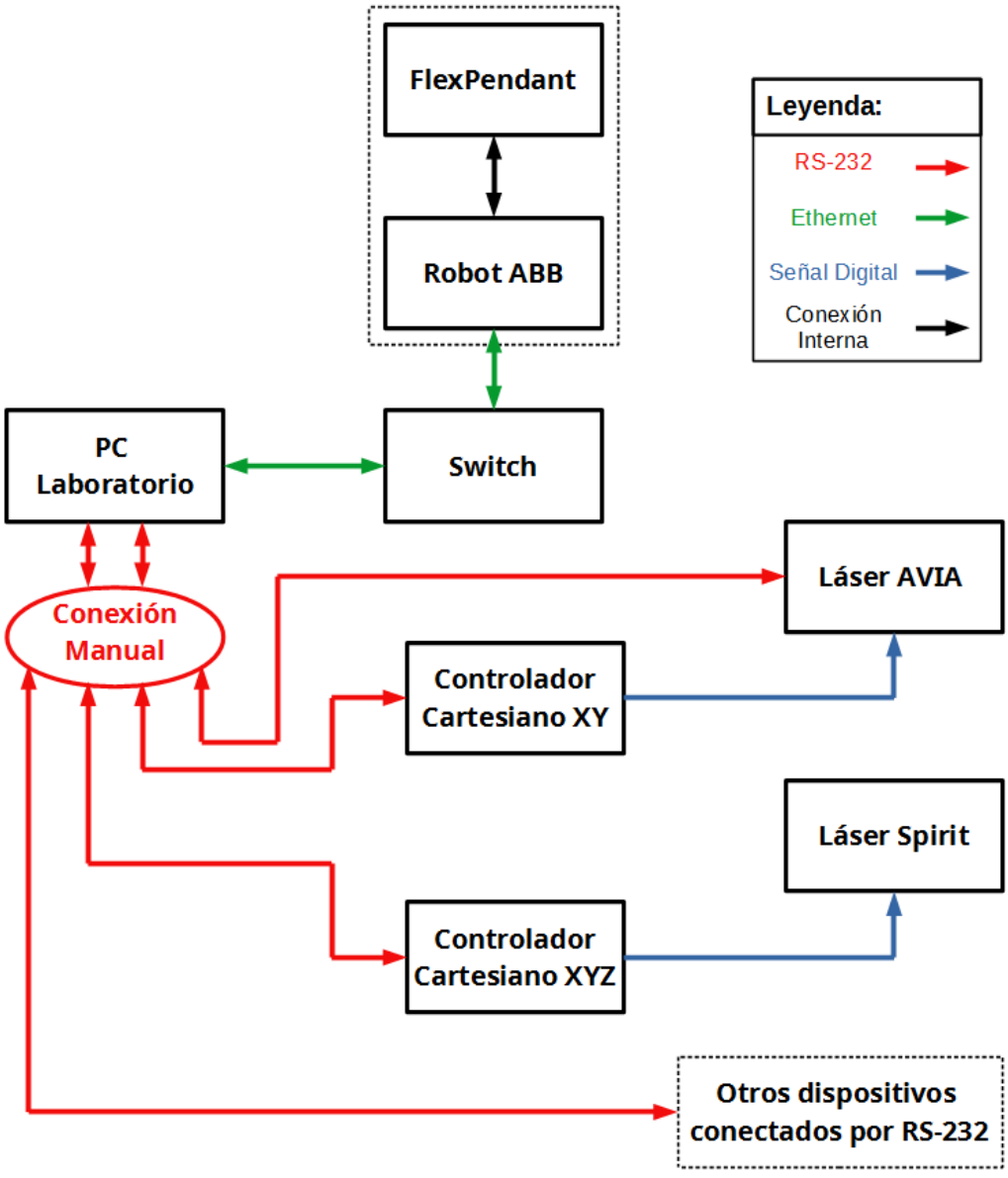


Figura 3.15 – Diagrama de comunicaciones inicial

Véase que las dobles o simples flechas indican bidireccionalidad o unidireccionalidad en la comunicación.

3.8.3. Carácterísticas de los Equipos

En este punto se describen las posibilidades de conectividad de los equipos, sus prestaciones de cara al proceso de micromecanizado y el interés que tiene su uso actualmente.

3.8.3.1. Láser New Port, Spectra Phisics, Spirit 1040-4

Se trata de un láser pulsado de alta frecuencia bombeado por diodos. Además del propio equipo láser dispone de un PC portátil adjuntado con el equipo por el fabricante. El PC corre una interface propietaria bajo Windows 7, que es necesaria para el manejo del equipo aunque se realice simultáneamente el control externo del mismo.

Clasificado según norma UNE-EN-60825-1, equivalente a la EIC60825 como láser de categoría 4, es decir, la máxima categoría de peligrosidad siendo muy peligrosas incluso radiaciones difusas.

Sus principales características son:

- Prestaciones

Tipo de pulsado	Mode locking
Potencia Media	$\geq$ 4w a 200,500,750 y 1000KHz de pulsado
Energía de pico	50μJ a 100KHz de pulsado
PRF(Pulse Repetition Frecuency)	Mínima: 1 único pulso Máxima: 1000KHz
Ancho de pulso (duración)	Mínimo: 300fs Máximo: Máximo nominal

Tabla 3.4 – Prestaciones Pulsado Láser Spirit

Longitud de onda	1040$\pm$5nm
Diámetro del haz	1,5mm
Polarizado	Horizontal

Tabla 3.5 – Prestaciones Haz Láser Spirit

- Conectividad

  - **User Control Interface:** Así es como referencia el fabricante a una serie de conexiones analógicas y digitales aunadas en un conector común del tipo SubD 21WA4 (M).

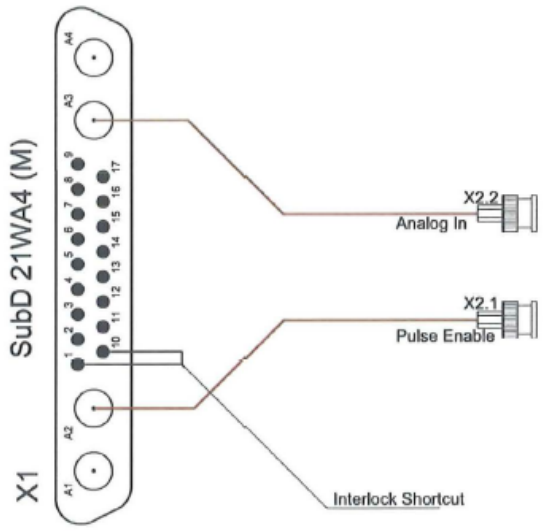


Figura 3.16 – Conector Láser Spirit

- **Pin A2:** Pulse Enable (Conector BNC): Señal digital de entrada TTL (0-5V) activa a nivel alto (5V). Tiene instalado un adaptador a conector BCN.
- **Pin A3:** Analog In (Conector BNC): Señal analógica de entrada de modulación de potencia. Resolución máxima 12bits correspondiente a 0-5V Tiene instalado un adaptador a conector BCN.
- **Pin 1:** I-Lock +: Pin positivo para interlock, dispositivo de seguridad basado en contactos en serie normalmente cerrados.
- **Pin 2:** Custom Error: Pin para señal digital de salida asignable a una fuente de error del equipo.
- **Pin 3:** 5V (1ª max.): Fuente de alimentación.
- **Pin 4:** GND: Referencia de masa de los pines intermedios, es decir, los que no disponen de masa propia.
- **Pin 10:** I-Lock -: Pin negativo para interlock, dispositivo de seguridad basado en contactos en serie normalmente cerrados.
- **Pin 11:** Custom Error: Pin para señal digital de salida asignable a una fuente de error del equipo.
- **Otros pines:** No conectados.

- **Conector CANopen:** Conector para comunicar el equipo a través del protocolo CANopen el cual está basado en CAN bus.

- **Interés para el proceso:** Tiene un interés primario dado que es uno de los equipos principales necesarios para micromecanizado.

3.8.3.2. Láser Coherent, AVIA Ultra 355-20008

Se trata de un láser pulsado de alta frecuencia bombeado por diodos y de estado sólido.

Clasificado según norma UNE-EN-60825-1, equivalente a la EIC60825 como láser de categoría 4, es decir, la máxima categoría de peligrosidad siendo muy peligrosas incluso radiaciones difusas.

Sus principales características son:

- Prestaciones

Tipo de pulsado	Q-Switched
Potencia Media	2,0w a 30KHz de pulsado 1,0 w a 60KHz de pulsado
PRF(Pulse Repetition Frecuency)	Mínima 1 único pulso Máxima 100KHz
Ancho de pulso (duración)	Mínimo: $\leq$300fs Máximo: 30KHz

Tabla 3.6 – Prestaciones Pulsado Láser AVIA

Longitud de onda	355nm
Diámetro del haz	2,2mm $\leq$10 %
Polarizado	Vertical

Tabla 3.7 – Prestaciones Haz Láser Avia

- Conectividad
  - **PULSE TRIGGER IN (Conector BNC):** Entrada, señal digital de disparo, inicia el pulsado.
  - **PULSE SYNC OUT (Conector BNC):** Salida, señal digital de disparo, sirve para sincronizar otros equipos con el disparo.
  - **EXTERNAL ENABLE (Conector BNC):** Entrada, señal digital, resistencia pull-up interna que debe ser puesta a masa para disparo.
  - **EXTERNAL INTERFACE:** Una serie de pines digitales de entrada y salida para monitorizar alertas y manejar parámetros del disparo.
  - **OPTIONS CONECTOR:** Una serie de pines digitales de entrada y salida para monitorizar y manejar el disparo.
  - **EXTERNAL INTERLOCK:** Conector de seguridad normalmente abierto, debe estar cerrado para permitir el disparo. Se suele conectar a elementos de seguridad como finales de carrera en las puertas del laboratorio.
  - **Puerto serie RS-232 (Conector DB-9):** Permite el control remoto del equipo mediante una serie de comandos.
- **Interés para el proceso:** Tiene un interés primario dado que es uno de los equipos principales necesarios para micromecanizado.

3.8.3.3. Brazo Robot ABB, IRB 120 integrado junto a controladora ABB, IRC5 Compact

El equipo se compone de un brazo robot ABB y su controladora montados sobre un banco de trabajo móvil, en tal forma que se puede desplazar el conjunto fácilmente y fijarse mediante unas trabas del banco móvil para mantener la posición del conjunto estática respecto al resto de equipos.

A continuación se describen las características principales de los distintos elementos del conjunto:

- Brazo ABB, IRB 120
  - Características
    - Antropomórfico
    - Manipulador
    - 6 GDL
    - Servocontrolado
    - Dispone de frenos en los ejes
    - Protección IP30
    - Entradas/Salidas digitales y de aire comprimido en el elemento terminal
  - Prestaciones
    - Peso: 25Kg
    - Carga máxima: 3Kg
    - Alcance máximo: 0,58m
    - Tomas neumáticas a presión de 5bar máximo
    - Rango de trabajo: Ver la imagen de abajo

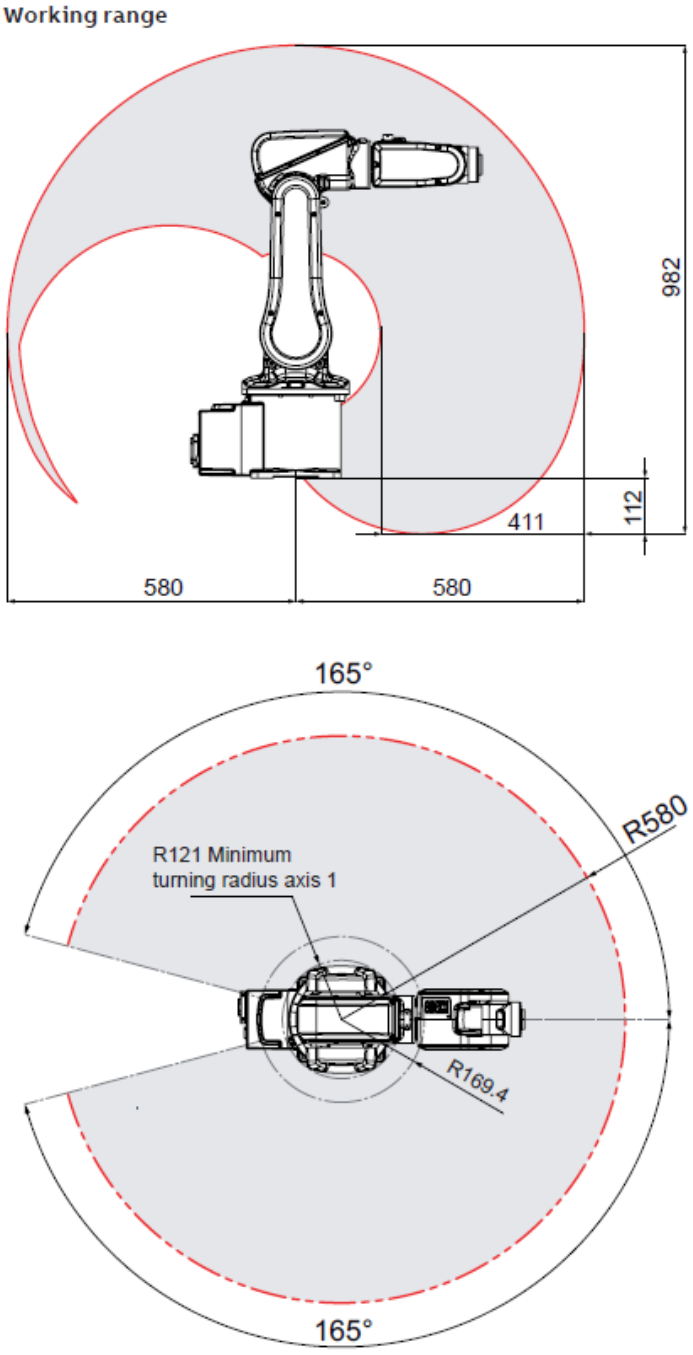


Figura 3.17 – Volumen de trabajo ABB, IRB 120

- Controladora ABB, IRC5 Compact
  - Características
    - Integra: Computadora, Electrónica de potencia, Consola ABB FlexPendant, Control electrónico e interfaces de comunicación y automatización.
    - Sistema operativo: RobotWare
    - Programación: A través de Flex Pendante o con un PC adicional empleando el software RobotStudio de ABB. Se programa en el lenguaje de programación

RAPID de ABB.
    - Peso: 50Kg
- Conectividad
    - Conectores RJ-45 Ethernet: Dispone de 7 conectores de este tipo, cada uno tiene una función predeterminada como actualización de software, mantenimiento... Uno de ellos es el que se emplea para programar el equipo de forma remota y así se referencia en los diagramas.
    - USB
    - Conectores DB-9: Uno es COM1 para comunicación serie RS-232 y el otro está pensado para fines de depuración.
    - Conector lateral DB-9(XS-18): ProfiBus.
    - Conectores tipo borna:
        - De seguridad: Tres conectores de múltiples bornas cada uno (XS7, XS8 Y XS9) para las funciones de: paro automático, paro general, paro de emergencia externo, alimentación externa, PTC externo.
        - 16 salidas digitales optoacopladas (0-24V) (XS14 y XS15)
        - Fuente de alimentación (XS16): de tensión DC 24V y corriente $\leq$6A.
        - Bus de Campo DeviceNet (XS17)

- **Interés para el proceso del conjunto:** Tiene un interés primario dado que es uno de los equipos principales necesarios para micromecanizado, es el único sistema de translación del laboratorio que permite además del desplazamiento de la pieza de trabajo, su orientación.

3.8.3.4. PC laboratorio

Este PC está instalado en el laboratorio y es el equipo que se empleaba principalmente para programar, configurar y monitorizar los distintos equipos. Dispone de dos tarjetas Ethernet conector RJ-45 y dos puertos serie conector DB-9. Su hardware está obsoleto y no permite un manejo fluido de los equipos. Por otro lado, el hecho de que su número de puertos es limitado y que se podría considerar el nodo central de la red de comunicaciones ocasiona que el trabajo con distintos equipos y periféricos sea tedioso teniendo que conectar y desconectar cableado según la aplicación. Es un claro cuello de botella.

3.8.3.5. Switch Ethernet

Se encuentra instalado en la misma mesa que el PC del laboratorio. Permite la conexión de múltiples equipos a través de cable Ethernet empleando protocolos de la familia TCP/IP.

3.8.3.6. Controladores Cartesianos

Estos equipos de la casa Newport se conectan al PC del laboratorio a través de conexión por puerto serie RS-232. Eran empleados para micromecanizado.

Como indica el título son sistemas de translación y posicionamiento cartesianos de uno y dos grados de libertad respectivamente correspondientes con los ejes X, Y e X, Y, Z. Eran los únicos sistemas de translación disponibles en el laboratorio antes de la adquisición del brazo robot. Actualmente su interés es secundario dado que por el limitado número de grado de libertad no permiten la rotación de la pieza de trabajo, por lo que no son adecuados para los procesos de micromecanizado que son de interés actualmente.

3.8.3.7. Otros Periféricos RS-232

Se dispone de una serie de periféricos y sensores con conectividad RS-232. Estos equipos son cámaras, escáneres y demás sensores. El interés en estos equipos es relativo, no es prioritario, pero si interesa disponer de un método de conexión a estos equipos sin que se formen cuellos de botella.

4 NORMAS Y REFERENCIAS

4.1. Disposiciones legales y normas aplicadas

- *UNE-EN 60825-1*. Seguridad de los productos láser. Parte 1: Clasificación de los equipos y requisitos. Génova, 6 2804 MADRID-España: AENOR. 2008.
- *UNE-EN 60825-4*. Seguridad de los productos láser. Parte 4: Sistemas de protección frente a la radiación láser. Génova, 6 2804 MADRID-España: AENOR. 2007.
- *UNE-EN 11553*. Seguridad de las máquinas. Máquinas de procesamiento láser. Génova, 6 2804 MADRID-España: AENOR. 2009.
- *UNE 1027*. Dibujos Técnicos. Plegado de Planos.
- *UNE 1032*. Dibujos Técnicos. Principios Generales de Representación.
- *UNE 1039*. Dibujos Técnicos. Acotación.
- *UNE-EN ISO 3098-0*. Documentación técnica de productos. Escritura. Requisitos generales.
- *UNE-EN ISO 5455*. Dibujos Técnicos. Escalas.
- *UNE-EN ISO 5457*. Documentación técnica de productos. Formatos y representación de los elementos gráficos de las hojas de dibujo.
- *UNE-EN ISO 7200*. Documentación técnica de productos. Campos de datos en bloques de títulos y en cabeceras de documentos.

Además, se cumplirá la normativa establecida por la *Escuela Universitaria Politécnica Ferrol* para la elaboración de los TFG que se aplica a las titulación de Grado en Ingeniería Electrónica y Automática.

4.2. Bibliografía

[1] ALONSO FERNANDEZ, BENJAMÍN; BORREGO VARILLAS, ROCÍO; HERNÁNDEZ GARCÍA, CARLOS; PÉREZ HERNÁNDEZ, J. ANTONIO; VÁZQUEZ ROMERO, CAROLINA; *El láser. La luz de nuestro tiempo*, 1ª ed. Salamanca: Isabel Arias Tabalina, 2010.

[2] VANDERPLAS, JAKE; *A Whirlwind Tour of Python*, 1ª ed. 1005 Gravenstein Highway North, Sebastopol, CA 95472 USA: O'REILLY, 2016.

[3] DAWSON, MICHAEL; *Python Programming for the Absolute Beginner*, 3ª ed. 20 Channnel Center Street Boston, MA 02210 USA: Course Technology, 2010.

[4] GRAYSON, JOHN E.; *Python and Tkinter Programming.*, 1ª ed. 32 Lafayette Place Greenwich, CT 06830 USA: Manning Publications Co., 2000.

4.3. Software utilizado

- ARDUINO. Arduino IDE. 1.8.9
- AUTODESK. AutoCAD 2019
- AUTODESK. Inventor 2019
- ULTIMAKER. Cura. Versión 3.6.0
- MICROSOFT. Excel. Versión 2013
- HW-GROUP. Hercules setup utility. Version 3.2.8
- KICAD. KiCAd. Versión 5.1.2
- MATHWORKS. MATLAB. Versión R2016a
- CADENCE. OrCAD Capture CIS - Lite. Versión 16.6-p005
- ABB. RobotStudio. Versión 6.08.01
- SPYDER-IDE. Spyder. Versión 3.3.2 (Python 3.7.1 64-bit)
- WIZNET. WIZnet-S2E-Tool-GUI. Versión 0.5.5 Beta

4.4. Otras referencias

[5] LAMAS VIGO, JAVIER;*Sistema de posicionamiento, automatización, muestreo y control para la fabricación en tres dimensiones utilizando el láser como herramienta.* Dirigido por Alberto Ramil Rego. Ferrol: Universidade da Coruña, 2015. Tesis doctoral: Programa de doctorado en Ingeniería Industrial.

[6] ABB;*Manual del producto IRB 120.* Se-721 68 Västerås, Suecia: AB, Robotics and Motion, 2018.

[7] ABB;*Manual del producto IRC5 Compact.* Se-721 68 Västerås, Suecia: AB, Robotics and Motion, 2018.

[8] ABB;*Operating manual IRC5 Integrator's guide.* Se-721 68 Västerås, Suecia: AB, Robotics and Motion, 2018.

[9] ABB;*Especificaciones del producto IRB 120.* Se-721 68 Västerås, Suecia: AB, Robotics and Motion, 2018.

[10] ABB;*Especificaciones del producto Controller IRC5 with FlexPendant.* Se-721 68 Västerås, Suecia: AB, Robotics and Motion, 2013.

[11] ABB;*Manual de refencia técnica Instrucciones, funciones y tipos de datos de RAPID.* Se-721 68 Västerås, Suecia: AB, Robotics and Motion, 2018.

[12] ABB;*Manual de referencia técnica Descripción general de RAPID.* Operator's Manual AVIA Ultra 355-2000S Laser.

[13] COHERENT;*Operator's Manual AVIA Ultra 355-2000S Laser.* 5100 Patrick Henry Drive Santa Clara, CA 95054, USA: COHERENT.

[14] SPECTRA-PHYSICS(A NEWPORT CORPORATION BRAND);*Spirit 1040-4 Spirit 1040-4-SHG.* 3635 Peterson Way Santa Clara, CA 95054, USA: Spectra-Physics(A Newport Corporation Brand).

[15] *National Instruments Corporation*;*¿Qué es Adquisición de Datos?*[en línea]. National Instruments,[Fecha de consulta: 1 de mayo del 2019]. Disponible en: https://www.ni.com/data-acquisition/what-is/esa/

[16] *Centro de Láseres Pulsados (CLPU).¿Qué son la emisión espontánea, la emisión estimulada y la inversión de población?* [en línea]. CLPU. [Fecha de consulta: 16 de Abril del 2019]. Disponible en: :https://www.clpu.es/es/divulgacion/bits/que-son-la-emision-espontanea-la-emision-estimulada-y-la-inversion-de-poblacion

[17] *Wikimedia.Comunicación Serie.* [en línea]. Wikipedia, the free encyclopedia. [Fecha de consulta: 1 de mayo del 2019]. Disponible en: https://es.wikipedia.org/wiki/Comunicaci%C3%B3n_serie

[18] *Wikimedia.DeviceNet.* [en línea]. Wikipedia, the free encyclopedia [Fecha de consulta: 2 de mayo del 2019]. Disponible en: https://es.wikipedia.org/wiki/DeviceNet

[19] *Wikimedia.Espectro electromagnético.* [en línea]. Wikipedia, the free encyclopedia [Fecha de consulta: 27 de mayo del 2019]. Disponible en: https://es.wikipedia.org/wiki/Espectro_electromagn%C3%A9tico

[20] *Wikimedia.Física/Física moderna/Efecto fotoeléctrico.* [en línea].Wikibooks. [Fecha de consulta: 16 de abril del 2019]. Disponible en: https://es.wikibooks.org/wiki/F%C3%ADsica/F%C3%ADsica_moderna/Efecto_fotoel%C3%A9ctrico

[21] *Wikimedia*. *Láser.* [en línea]. Wikipedia, the free encyclopedia. [Fecha de consulta: 16 de abril del 2019]. Disponible en: https://es.wikipedia.org/wiki/Espectro_electromagn%C3%A9tico

[22] *Jaime Caballero*. *Lente convergente: características, tipos y ejercicio resuelto.* .[en línea]. Lifeder. [Fecha de consulta: 1 de mayo del 2019]. Disponible en: https://www.lifeder.com/lente-convergente/

[23] *Wikimedia*. *Luz*. [en línea]. Wikipedia, the free encyclopedia. [Fecha de consulta: 8 de abril del 2019]. Disponible en: https://es.wikipedia.org/wiki/Espectro_electromagn%C3%A9tico

[24] *Wikimedia*. *Microcontrolador.* [en línea]. Wikipedia, the free encyclopedia. [Fecha de consulta: 2 de mayo del 2019]. Disponible en:https://es.wikipedia.org/wiki/Microcontrolador

[25] *Wikimedia*. *Protocolo de control de transmisión.* [en línea]. Wikipedia, the free encyclopedia. [Fecha de consulta: 2 de mayo del 2019]. Disponible en: https://es.wikipedia.org/wiki/Protocolo_de_control_de_transmisi%C3%B3n

[26] *Wikimedia*. *Protocolo de datagramas de usuario.* [en línea]. Wikipedia, the free encyclopedia. [Fecha de consulta: 2 de mayo del 2019]. Disponible en: https://es.wikipedia.org/wiki/Protocolo_de_datagramas_de_usuario

[27] *Wikimedia*. *Puerto serie.* [en línea]. Wikipedia, the free encyclopedia. [Fecha de consulta: 1 de mayo del 2019]. Disponible en: https://es.wikipedia.org/wiki/Puerto_serie

[28] *Wikimedia*. *RS-232.* [en línea]. Wikipedia, the free encyclopedia. [Fecha de consulta: 1 de mayo del 2019]. Disponible en: https://es.wikipedia.org/wiki/RS-232

[29] *Wikimedia.Universal Asynchronous Receiver-Transmitter.* [en línea]. Wikipedia, the free encyclopedia. [Fecha de consulta: 1 de mayo del 2019]. Disponible en: https://es.wikipedia.org/wiki/Universal_Asynchronous_Receiver-Transmitter

[30] *Jacob Beningo*. *USART vs UART: Know the difference.* .[en línea]. EDN Network. [Fecha de publicación: 21 de septiembre del 2015] [Fecha de consulta: 1 de mayo del 2019].

Disponible en: https://www.edn.com/electronics-blogs/embedded-basics/4440395/USART-vs-UART--Know-the-difference

[31] *WIZnet.WIZ750SR-110.* [en línea]. WIZnet. [Fecha de consulta: 13 de mayo del 2019]. Disponible en: https://www.wiznet.io/product-item/wiz750sr-110/

[32] *Doug Hellman.Doug Hellman.* [en línea]. PyMOTW. [Fecha de publicación: 16 de marzo del 2019] [Fecha de consulta: 17 de marzo del 2019] Disponible en: https://pymotw.com/2/socket/tcp.html

5 DEFINICIONES Y ABREVIATURAS

- **3D:** Tres Dimensiones.
- **AD/DA:** Convertidor AD es un dispositivo que digitaliza señales analógicas. Ánalogamente un convertidor DA es un dispositivo que reconstrulye señales digitales en su forma analógica.
- **ASCI:** American Standard Code for Information.
- **BNC:** Conector del tipo Bayonet Neil-Concelman.
- **CIT:** Centro de Investigacións Tecnolóxicas de Ferrol.
- **CPU:** Central Processing Unit.
- **DAQ:** Data AcQuisition.
- **DC:** Direct Current.
- **GDL:** Grados De Libertad.
- **Láser:** Light Amplification by Stimulated Emision of Radiation.
- **LCD:** Liquid Cristal Display.
- **LED:** Light-Emitting Diode.
- **PC:** Personal Computer.
- **PCB:** Printed Circuit Board.
- **PLC:** Programmable Logic Controller.
- **PRF:** Pulse Repetition Frecuency.

- **PWM:** Pulse Width Modulation.
- **RAM:** Random Accsess Memory.
- **ROM:** Read-Only Memory.
- **RS-232:** Recommended Standard 232.
- **TCP:** En este trabajo aparece con dos significados: Tool Central Point cuando se habla de la herramienta del brazo robot y Transmision Control Protocol en relación a redes de comunicación.
- **TFG:** Trabajo Fin de Grado.
- **TTL:** Transistor-Transistor Logic.
- **WIFI:** Wireless Ethernet Compatibillity Alliance.

6 REQUISITOS DE DISEÑO

En este capítulo de la memoria se presentan las bases y datos de partida establecidos por el tutor, así como las interfaces con otros sistemas o elementos externos al TFG u otros que condicionan las soluciones técnicas del mismo. Estos requisitos y restricciones se exponen a continuación:

- Se deberá aportar una solución para adaptar las comunicaciones punto a punto RS-232 entre los distintos equipos y el PC del laboratorio para que sean multipunto y accesibles desde la red local ethernet del laboratorio. Deberá ser así para que distintos equipos puedan comunicarse sin necesidad de modificar cableado.
- Se deberá aportar una solución para el disparo de los equipos láser y modulación de su potencia empleando las interfaces que se consideren más adecuadas a tal propósito.
- Se deberán integrar todos los equipos que se necesite programar o configurar y monitorizar sus procesos en una única red, preferiblemente bajo protocolos de la familia TCP/IP y con un punto de acceso WiFi.
- Las soluciones de Hardware y montajes definitivos deben estar conformados y cableados de manera suficientemente robusta de tal forma que en la medida de lo posible no se produzcan fallas y puedan ser usados de forma segura por el personal del laboratorio.
- Se deberá asegurar que las prestaciones del conjunto permiten coordinar la acción de los equipos láser con el posicionamiento de la pieza de trabajo a través del sistema de translación (brazo robot).

- Las soluciones aportadas, tanto hardware como software, serán flexibles, sencillas y actualizables en la medida de lo posible, para en un futuro pudieran ser adaptadas a distintos propósitos. El objetivo es que los equipos instalados en el laboratorio sean lo suficientemente versátiles para que se facilite la labor investigadora.
- Se deberá dejar el trabajo realizado bien documentado para que un tercero pudiera mantener y hacer uso de los equipos con la mínima dificultad.

7 ANÁLISIS DE LAS SOLUCIONES

7.1. Posibilidades de distribución de los equipos

Los equipos de mayor interés para el proceso de micromecanizado son los láseres pulsados Spirit y AVIA y el brazo robot ABB. Este último constituye el sistema de translación y posicionamiento de las piezas de trabajo para el mecanizado con ambos láseres.

Hay que tener en cuenta que cualquier colocación del brazo robot en el laboratorio nunca es definitiva, dado que se encuentra montado sobre un banco de trabajo con ruedas que permiten su colocación en cualquier lugar y sin necesidad de desmontar nada. Además dispone de unos bloqueos en las ruedas para que, una vez decidida su posición, se pueda fijar y no se produzcan errores de calibración debidos a esta característica.

Para decidir la distribución de los equipos, se establecen las siguientes pautas de interés:

- Posibilidad de trabajar con ambos láseres sin cambiar conexiones.
- Posibilidad de trabajar con ambos láseres sin desplazar los equipos de posición en el laboratorio.
- Distribución que minimice las reflexiones peligrosas hacia el personal que maneja los equipos. Especialmente en los lugares posteriores a donde el láser atraviesa las lentes.
- Posibilidad de instalar pantallas de protección conforme a la norma UNE-EN 60825.
- Facilidad en cuanto a mantenimiento de los equipos y modificación del sistema.
- Buen aprovechamiento de las características dinámicas, de resolución y de alcance del sistema de translación.
- Empleo del mínimo número de espejos para reconducir los haces hacia la zona de mecanizado.
- Que la distancia entre la salida del haz la zona de mecanizado sea mínima.

- Optimización del espacio en el laboratorio.

- Los equipos láser son muy costosos de mover, por lo que si es posible deben conservar su posición en la solución aportada.

La disposición inicial de los equipos es la de la siguiente imagen:

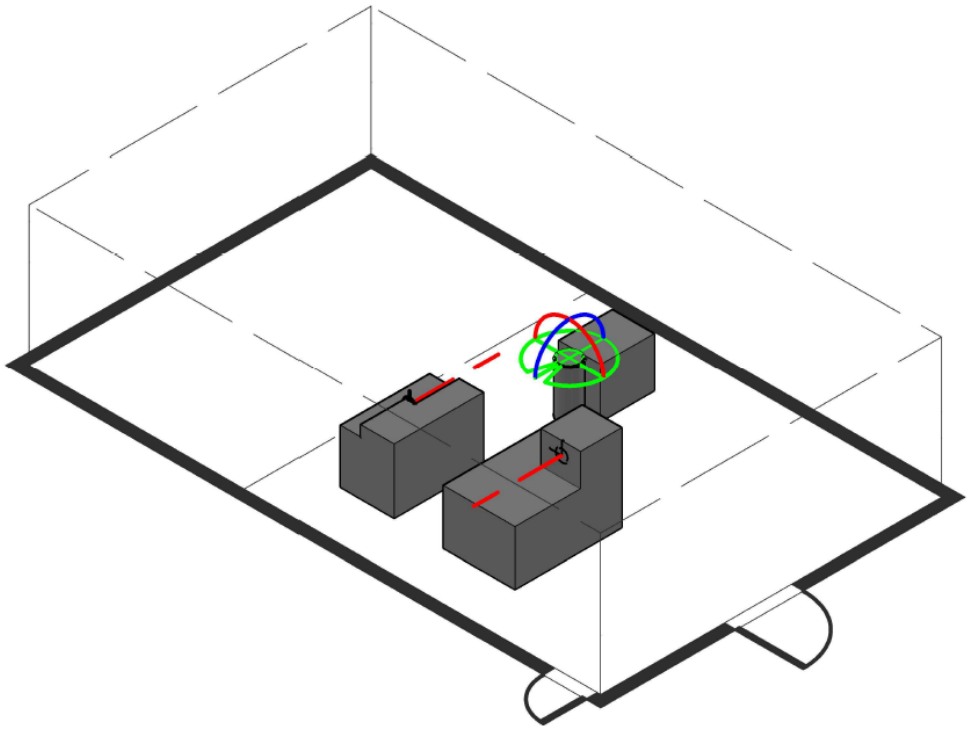


Figura 7.1 – Distribución inicial de los equipos

En la imagen se puede ver un croquis de los equipos en su posición inicial en el laboratorio. Los equipos que se representan en la imagen de arriba abajo y de izquierda a derecha son el brazo robot junto a el volumen de trabajo encerrado por el eje de su muñeca, el láser Spirit (que apunta hacia el brazo) y el láser AVIA (que apunta en sentido opuesto al robot).

En ambos equipos láser se representa el haz laser el linea discontínua roja. En cuanto al volumen de trabajo del brazo robot, es mas o menos una semiesfera que se representa como sus secciones con los planos normales a las direcciones de los ejes cartesianos del brazo.

El problema de la distribución de puede resumir en cómo colocar el brazo robot y cuantos espejos y donde se colocan para direcciónar los haces. Teniendo en cuenta que en laboratorio existe más equipamiento y mobiliario, y que no se deben bloquear los accesos se puede acotar su superficie de la siguiente manera.

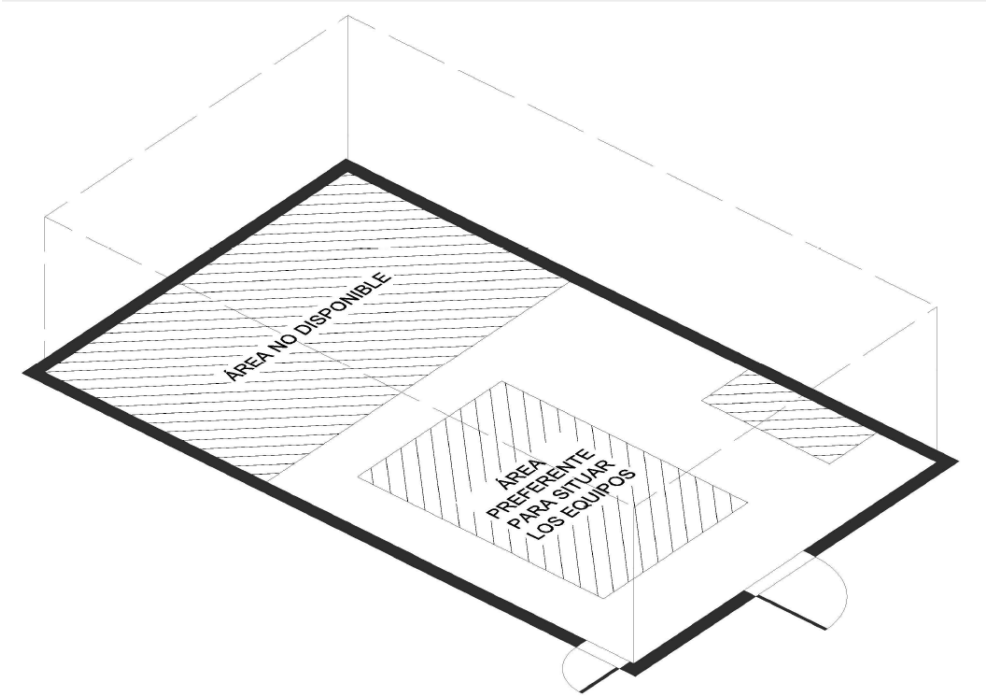


Figura 7.2 – Áreas delimitadas

Se distinguen dos áreas una periférica formada por dos rectángulos que están pegados a las paredes del laboratorio y un área formada por un rectángulo central. El área periférica está ocupada por otros equipos y mobiliario. Para una distribución óptima el área sin marcar debe quedar libre y el área central es donde se deben situar los equipos.

Las dimensiones principales son las siguientes (las posiciones están tomadas con respecto a la esquina interior de las paredes que tienen puerta en el plano del suelo):

- Laboratorio:
  - Dimensiones: 577x965x236cm
  - Área óptima equipos:
    - Dimensiones: 275x403cm
    - Posición: 82x120cm
- Láser AVIA:

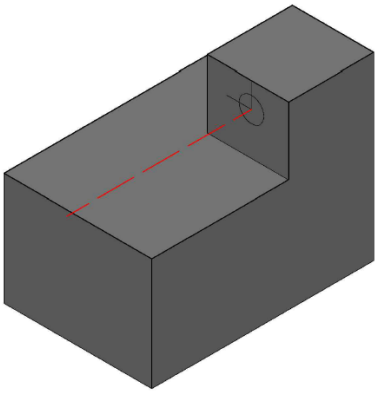


Figura 7.3 – Láser AVIA

- Posición: 820x2770mm
- Dimensiones mesa: 1800x1200x910mm
- Haz:
  - Posición de salida: 1930x2560mm
  - Altura respecto a suelo: 1260mm

- Láser Spirit:

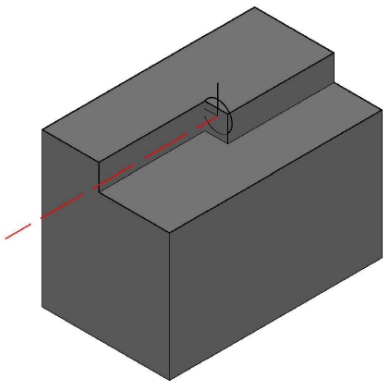


Figura 7.4 – Láser Spirit

- Posición: 840x4330mm
- Dimensiones mesa: 1500x900x900mm
- Haz:
  - Posición de salida: 1580x4830mm
  - Altura respecto a suelo: 1040mm
  - Altura obstáculo respecto mesa: 180mm El obstáculo es la parte del equipo que impide direccionar el haz hacia un punto medio entre los haces de ambos láseres.

- Brazo Robot:

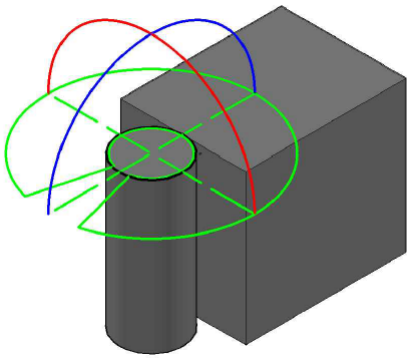


Figura 7.5 – Brazo Robot

- Dimensiones mesa: 900x700x840mm
- Dimensiones cilindro soporte brazo:
  - Radio: 180mm
  - Altura: 840mm
- Volumen de trabajo:

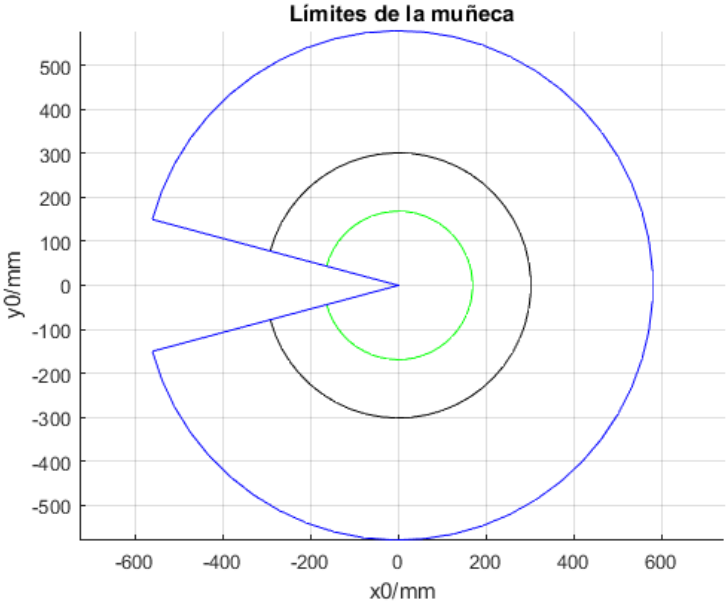


Figura 7.6 – Volumen de trabajo en planta

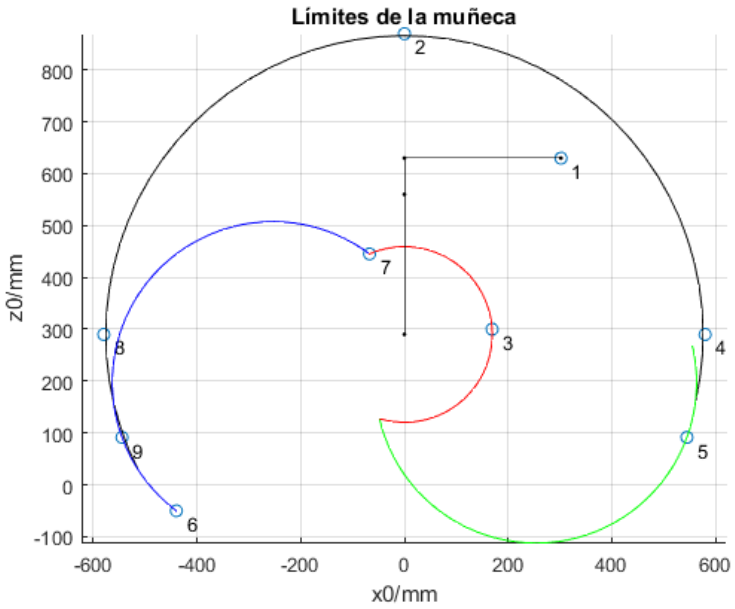


Figura 7.7 – Volumen de trabajo perfil

A partir de los datos de la imagen se considera el volumen de trabajo del brazo respecto al eje de la muñeca como una semiesfera de radio 580mm.

7.2. Soluciones de hardware

El objetivo es actualizar la red de comunicaciones manteniendo flexibilidad en el sistema (ya que es un laboratorio y no un proceso fijo), alcanzando a su vez las prestaciones necesarias que posibiliten el proceso de micromecanizado de superficies 3D.

La manera de trabajar con el brazo robot que plantea el fabricante es a través de programas que ejecuta el equipo, bien subiéndolos y ejecutándolos de forma independiente o bien con un PC conectado con el software Robot Studio.

El flujo de trabajo más adecuado consiste en cargar un programa en el brazo, y luego ejecutarlo desde este de forma independiente. Esto es así por dos razones.

En primer lugar, el formato del archivo de programación es un archivo de texto, como las operaciones a realizar son siempre las mismas es posible implementar las trayectorias desde un software independiente del fabricante, que sea libre, y que permita una programación de las trayectorias más sencilla en tanto a las restricciones que se han de cumplir en este proceso. De manera mas concreta, una vez superada la fase de depuración en la implementación del proceso, las tareas a realizar con el brazo robot serán susceptibles de automatización. Por tanto ahorraría mucho trabajo el desarrollo de una aplicación para PC, que genere de forma automática el código en el lenguaje de programación del brazo robot. Lo haría en un archivo de texto que se podría cargar directamente en el brazo robot.

En segundo lugar, la controladora del brazo posibilita el manejo de salidas analógicas y digitales y buses de campo, muy adecuados para cumplir con los requerimientos de fiabilidad y temporales; en tanto a las señales de disparo de los láseres y de regulación de su potencia

durante el proceso.

Tomando esto en cuenta, se pasa a discutir las soluciones respecto a los siguientes puntos:

- **Actualización de los equipos comunicados a través de puerto serie:** En la red inicial de comunicaciones, donde todos los equipos comunicados a través del puerto serie se conectaban al PC del laboratorio, se producía un cuello de botella. Sólo se podían conectar dos equipos al pc del laboratorio, ya que el PC sólo disponía de dos puertos. Además esto ocasionaba tener que desconectar y conectar equipos en función de los que se desearan emplear.

  La solución más sencilla y adecuada para este caso es la instalación de pasarelas de comunicaciones que adapten la comunicación serie RS-232 a Ethernet TCP/IP. En tal forma que se permitan conexiones multipunto si se instala un switch.

  Esto es viable, ya que existen soluciones hardware de prestaciones suficientes, que posibilitan implementaciones sencillas a precios relativamente económicos, como para que se pueda instalar una pasarela para cada equipo con puerto serie.

  En el caso de este trabajo la solución propuesta por el tutor es el microcontrolador WIZ750SR-110 del fabricante WIZnet. Cómo esta solución es perfectamente válida y ya se había realizado la adquisición de los microcontroladores en el arranque del trabajo no se plantean otras soluciones.

Figura 7.8 – Microcontrolador WIZ750SR-110

Este microcontrolador actúa como pasarela, en tal forma que lo que se escribe por su puerto serie lo escribe a través de ethernet mediante un protocolo TCP/IP que es configurable. Se comprobó empiricamente que el retardo temporal es que introduce en las señal (es decir, el paso de un protocolo a otro), respecto una comunicación directa es despreciable para las aplicaciones planteadas en este trabajo.

Las características del miccrocontrolador son las siguientes:

Tensión alimentación	5V DC (<110mA)
Dimensiones PCB	62x40mm
Conector alimentación	Jack DC macho interior 2,1mm positivo

Tabla 7.1 – Prestaciones WIZ750SR-110

Conector RS-232	DB-9 Macho
Señales disponibles	TXD (Pin3), RXD (Pin2), RTS, CTS, GND
Bits de dato	7 u 8 bits configurable
Velocidad (baudios)	9600 hasta 230000

Tabla 7.2 – Comunicación serie WIZ750SR-110

Conector ethernet	RJ-45(Ethernet Connector)
Protocolos TCP/IP	Servidor TCP, Cliente TCP, UDP (también puede emular puerto serie virtual)

Tabla 7.3 – Comunicación ethernet WIZ750SR-110

- **Posibilidad de acceso a la red con un PC externo:** La manera mas sencilla de resolver este punto es conectando un router wifi al switch al que se conectan los equipos. De esta manera se consigue tener acceso a todos los equipos desde un PC externo, el cual se conecta de forma inalámbrica a la red, de modo que no es necesario realizar conexiones. El hecho de poder conectarse desde un PC portátil a todos los equipos dinamiza en gran medida el trabajo porque posibilita programar este PC sin necesidad de estar en el laboratorio y una vez en el laboratorio, emplear este mismo PC como nodo que programa, controla y monitoriza los demás equipos de la red.

  Véase, que es necesario que esta red ethernet formada por el conjunto del switch, wifi y equipos conectados sea independiente las redes de la universidad para poder configurar sus parámetros de forma independiente y con la posibilidad de estar aislada de internet.

- **Control del disparo de los láseres y regulación de su potencia:** Con respecto a la comunicación con los láseres, hay dos parámetros con requerimientos especiales. Estos parámetros son el disparo y la modulación de la potencia. Los requerimientos son el tiempo de respuesta, fiabilidad flexibilidad.

  - Tiempo de respuesta: Lo ideal es que el tiempo de respuesta sea despreciable con respecto a la dinámica del proceso, o que por lo menos, sea mucho más rápido. También es importante que la respuesta se produzca en tiempo real, concretamente que se puedan coordinar la respuesta de estos equipos con las trayectorias realizadas por el sistema de translación de forma precisa.

- Fiabilidad: La importancia de estos parámetros es suficiente como para dedicar un canal de comunicaciones específicamente para cada uno en cada equipo.
- Flexibilidad: La solución aportada debe tener la capacidad de ser fácilmente modificable, por lo que en la solución aportada se debe evitar un exceso de complejidad técnica.

En la realización del proceso, desde el punto de vista del brazo robot se ejecuta un programa a través del cual se hace recorrer al brazo las trayectorias con entre otras características, velocidad constante.

Las trayectorias son el elemento más complejo a coordinar, por tanto la manera más sencilla de coordinar el disparo y la modulación de potencia en los láseres con estas es tomar como referencia temporal la ejecución del programa del brazo y a través de su controladora generar las señales para el control de los otros parámetros.

Respecto a esto, las posibles soluciones para la coordinación de estos parámetros son:

- Disparo: La interface de disparo de los láseres es un contacto BNC que funciona como trigger y es la única manera de dispararlos a los de forma externa de una manera similar. La controladora del brazo robot dispone de una tarjeta de salidas y entradas digitales, a través de la cual resulta sencillo generar la señal de disparo. El único problema es que estas salidas no son compatibles con las entradas de trigger de los láseres. El trigger del láser AVIA consta de un contacto eléctrico de 5V DC que se ha de cerrar para dispararlo, mientras que en el caso del láser Spirit se ha de poner a una entrada TTL a nivel alto 5V DC respecto a un pin de masa para dispararlo. Las salidas digitales de la controladora del brazo trabajan entre 0 y 24,5V. Respecto a las restricciones expuestas las soluciones válidas son el empleo de un relé en estado sólido, de un circuito con un semiconductor tipo mosfet que sirva de intreface de conmutación o mediante un optoacoplador a transistor. La mejor solución de estas es emplear un optoacoplador a transistor porque cumple con las restricciones, proporciona aislamiento galvánico (lo cual es deseable) y es económicamente más razonable que un relé en estado sólido.
- Regulación de la potencia: Se considera como solución más adecuada para este caso el empleo de una señal analógica generada a través de la controladora para regular la potencia. Esto es así porque se consideran como más importantes en un primer momento las características de secillez y flexibilidad. Este planteamiento no es realizable de forma inmediata debido a:
  - La controladora no dispone de salidas analógicas: En los manuales de referencia del conjunto del brazo robot y la controladora, el fabricante da a entender, de manera un poco ambigua que la controladora trae por defecto una tarjeta de salidas analógica conectada por el bus DeviceNet, pero esto no es así.
  - El láser AVIA no admite modulación analógica: Cabe destacar que la modulación de potencia en este equipo se realiza de forma indirecta. Se realiza cam-

biando los parámetros de configuración del laser que se pueden modificar mientras se dispara. Un ejemplo de estos parámetros es la corriente en los diodos que siver de fuente de bombeo de energía a la cavidad del láser.

Según estas restricciones se ha de tomar una solución distinta para cada equipo láser:

- Láser AVIA: La única manera de regular la potencia en este equipo es cambiando algunos de los parámetros de configuración que se pueden cambiar mientras dispara a través de comunicación RS-232. Como se expuso en puntos anteriores esta comunicación serie punto a punto se convertirá en multipunto a través de la instalación de pasarelas que adapten esta comunicación a protocolos de la familia TCP/IP mediante una red de ethernet. La solución ha de contemplar pues la comunicación a través de estes protocolos entre la controladora y el equipo láser para que se consiga de este modo coordinar la regulación de potencia con el resto del proeso.
- Láser Spirit: En este equipo se dispone de una entrada analógica de tensión específica para la modulación de la potencia. Esta entrada puede trabajar con distintos rangos de tensión a fondo de escala siendo de 0 a 5V la configuración de máxima resolución (12 bits). En cuanto a la generación de esta señal, de forma análoga a las otras soluciones planteadas se ha de generar desde la controladora del brazo robot, sin embargo esta no dispone de salidas analógicas. Las soluciones planteadas a esta problemática son las siguientes:
  1. Adquirir la tarjeta DAQ propuesta por el fabricante del brazo en la documentación del mismo. Es una apuesta segura pero tiene un elevado coste. Además los rangos de tensión de salida no coinciden con los de la entrada del láser.
  2. Adquisición de otro dispositivo que permita generar señales analógicas haciendo uso de algún bus compatible con la controladora. No es sencillo asegurar la compatibilidad y sencillez de implementación. Pese a que los buses empleados están perfectamente estandaizados el sistema operativo de la controladora del brazo es RobotWare de ABB y la compatibilidad puede dar problemas.
  3. Implementación de un dispositivo que se conecte a un bus de la controladora. La manera de realizar esta opción de manera razonablemente sencilla es adquiriendo un módulo de comunicación que permita adaptar un dispositivo lógico programable a uno de los buses. Sin embargo la documentación de estos módulos y su fiabilidad pueden llegar a ser un problema.
  4. Implementación de un dispositivo que a través de salidas digitales codifique la información analógica. Por ejemplo, un microcontrolador que lea el estado de 8 salidas digitales (8 bits 256 valores rango) y que genere una salida analógica cuyo valor sobre el fondo de escala pertinente sea proporcional

al número binario codificado en sus 8 entradas digitales.

5. Implementación de una DAQ conectada por RS-232 a la controladora. En este caso si existen módulos bien documentados que permiten adaptar la comunicación serie RS-232 a serie TTL por ejemplo. Las desventajas de esta opción son que requiere el diseño y fabricación del dispositivo y que en caso de avería o sustitución la flexibilidad queda comprometida. Una posible variante sería hacer uso de un PLC que disponga de salidas analógicas. De esta manera se facilitaría el mantenimiento y se aumentaría flexibilidad, pero eso sí, se aumentarían los costes, aparecería la necesidad del un software para programar el PLC, las capacidades del PLC estarían desaprobechadas, los niveles de tensiones podrían no coincidir y habría que ver si su respuesta temporal cumple con los requerimientos.

7.3. Soluciones de software

En este apartado se analizan las decisiones de carácter más global acerca de la forma de configurar los equipos y del software a emplear para llevar a cabo el proceso de micromecanizado que es el objetivo último.

Este análisis se estructura en base a tres puntos:

7.3.0.1. Lenguaje de programación de propósito general para PC

Por comodidad se decidió emplear un único lenguaje en el PC para programar las tareas de comunicación con los equipos de la red y llevar a cabo todas tareas de monitorización, programación y control que no dispusieran de un software cerrado. De este modo la cantidad de software de la que tiene que disponer un PC es mínima y el sistema se mantiene flexible.

En un primer momento los lenguajes barajados fueron Phyton y MATLAB. La razón de esta elección es que ambos lenguajes además de ser de uso muy generalizado tienen la característica común de ser interpretados. Es decir, posibilitan la ejecución de los códigos línea a línea facilitando en gran medida la depuración de los mismos, lo cual es ideal dada la variabilidad en los procesos llevados a cabo en las tareas de investigación. Otra característica común a ambas plataformas es el amplio número de librerías, especialmente las que facilitan las implementaciones de desarrollos matemáticos y las que generan elementos visuales como gráficas.

Finalmente, se escogió el lenguaje Python por los siguientes motivos:

1. La ejecución en el PC es mucho más ligera que MATLAB.
2. Python tiene una licencia de código abierto.

Cabe destacar que la programación en Python fue realizada empleando la versión Python 3 en lugar de Python2. Esto es así porque su sintaxis tiene diferencias y resulta tedioso adaptar programas de la versión 2 a la 3. Por eso mismo, se decidió emplear Python 3 ya que es la versión del lenguaje que se espera que tenga mayor continuidad en el tiempo.

7.3.1. Configuración de las pasarelas WIZ750SR-110

Estos microcontroladores actúan como pasarelas entre comunicación serie RS-232 y comunicación ethernet.

Del lado del puerto serie no hay mucho que configurar además de ciertos parámetros del protocolo, sin embargo del lado ethernet se pueden emplear distintos protocolos.

El fabricante Wiznet propone en la documentación del equipo tres maneras de programarlo:

1. **Puerto serie virtual (VSP):** Se hace a través de una aplicación que proporciona el fabricante y se ha de instalar en el PC. El software emula un puerto serie. En el sistema operativo pasarela en red se ve como si fuera un puerto serie integrado en la propia placa base del equipo. Tiene el inconveniente de que se ha de usar el software, que limita la conexión a un solo equipo y que no es adaptable a distintas plataformas.

2. **Protocolo UDP:** Se envían mensajes a una dirección en red determinada sin confirmar el correcto envío de estos. Únicamente pudiera llegar a ser interesante por que se pueden llegar a alcanzar tasas de transmisión mayores que con otros protocolos.

3. **Protocolo Servidor TCP:** Es el que resulta más interesante y que el fabricante recomienda, a diferencia de la configuración UDP se verifica el envío correcto de los datos, lo que se traduce en una mayor fiabilidad. El microcontrolador actúa como servidor.

4. **Protocolo cliente TCP:** Similar al anterior, con la diferencia de que el microcontrolador actúa como cliente.

7.3.2. Método de programación del brazo robot

El lenguaje de programación para el brazo robot (o lo que es lo mismo, para su controladora IRC5 Compact) es RAPID. Este lenguaje es desarrollado por ABB parala programación específica de estos equipos y de hecho la facilita en gran medida.

Típicamente para programar en este lenguaje se emplea el entorno de programación de ABB; Robot Studio. Este entorno incluye múltiples características como la simulación física en 3D del comportamiento del equipo, o la depuración del código.

Sin embargo Robot Studio es un software que no es de licencia libre y es relativamente pesado para los equipos. Por eso mismo se plantea la posibilidad de generar de scripts de forma automática en RAPID a través de Python. Eso si, siempre y cuando se precise la realización de tareas repetitivas o cuya geometría no se defina ”a mano”; sino por ejemplo a partir de unos datos que obtenidos a través de un escáner 3D.

7.4. Diagramas de posibles soluciones

El siguiente diagrama a priori es la mejor solución para las comunicaciones en el laboratorio, pero por los factores citados hubo que desecharla:

- Todos los equipos tienen la capacidad de comunicarse más o menos a través de uno o más estándares de bus de campo, pero no existe un estándar común a todos. Por lo que habría que recurrir a distintas pasarelas aumentando la complejidad y poniendo en compromiso la fiabilidad.
- El empleo de un estándar de bus de campo implica cerrarse a ese medio de comunicación con las limitaciones que ello conlleva. Por otro lado el mantenimiento y actualización de las interfaces del bus de campo es más complicado en el caso de que el mantenimiento, actualización y ampliación de la red lo tenga que hacer una persona no familiarizada con este tipo de comunicaciones industriales.
- En el laboratorio no hay requerimientos de compatibilidad electromagnética ni de inmunidad ante el ruido como un ambiente industrial. Estas prestaciones que típicamente tienen los buses de campo industrial no son estrictamente necesarias (por lo menos al mismo nivel).

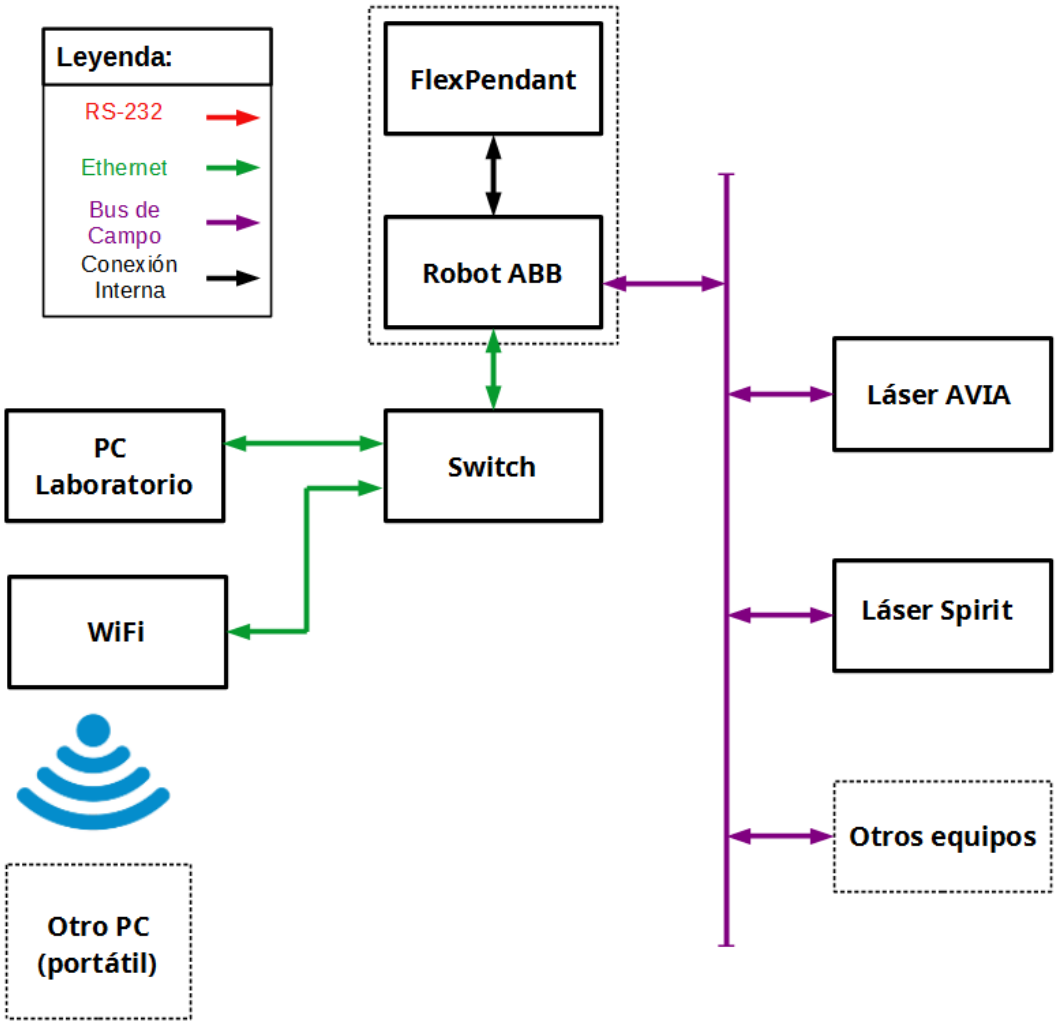


Figura 7.9 – Diagrama alternativo de la red del laboratorio

8 RESULTADOS FINALES

8.1. Distribución final de los equipos

Se estima como mejor distribución para los equipos la representada en las siguientes ilustraciones:

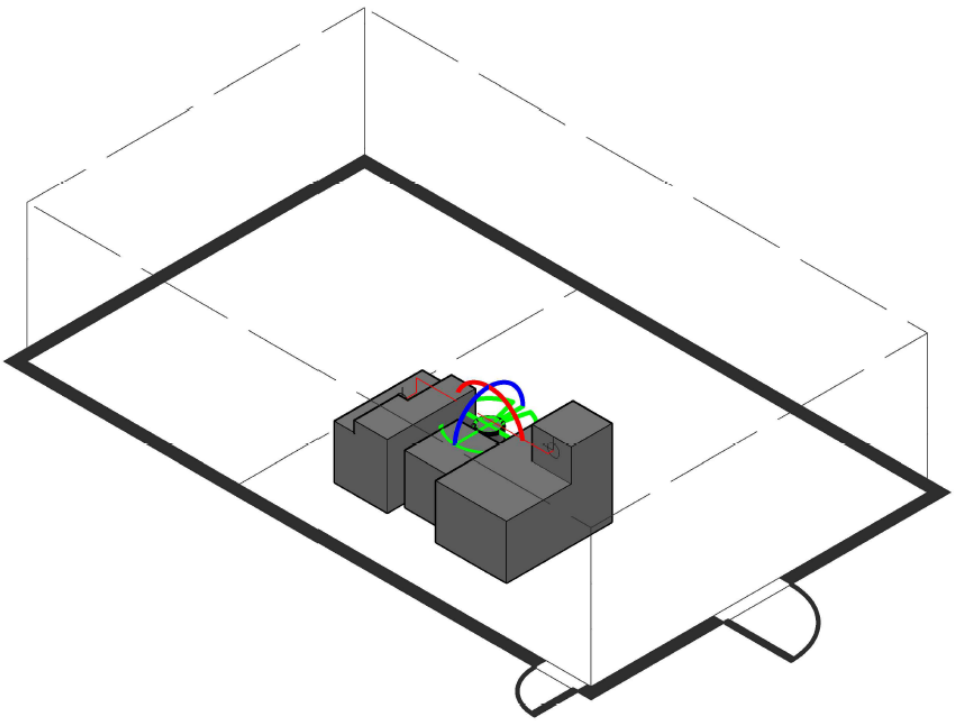


Figura 8.1 – Distribución final, perspectiva 1

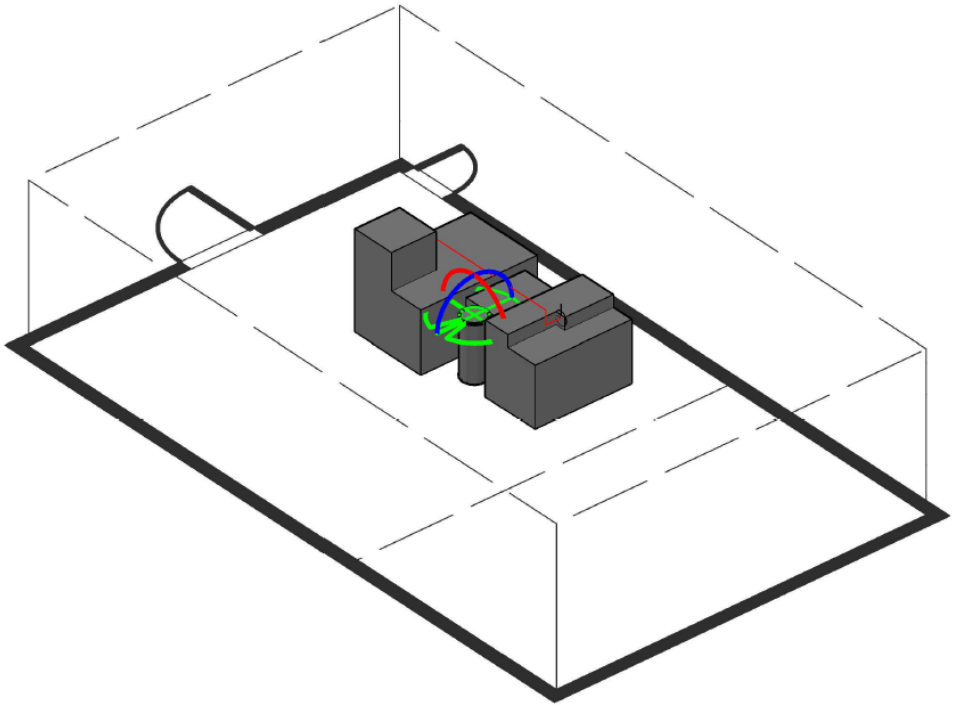


Figura 8.2 – Distribución final, perspectiva 2

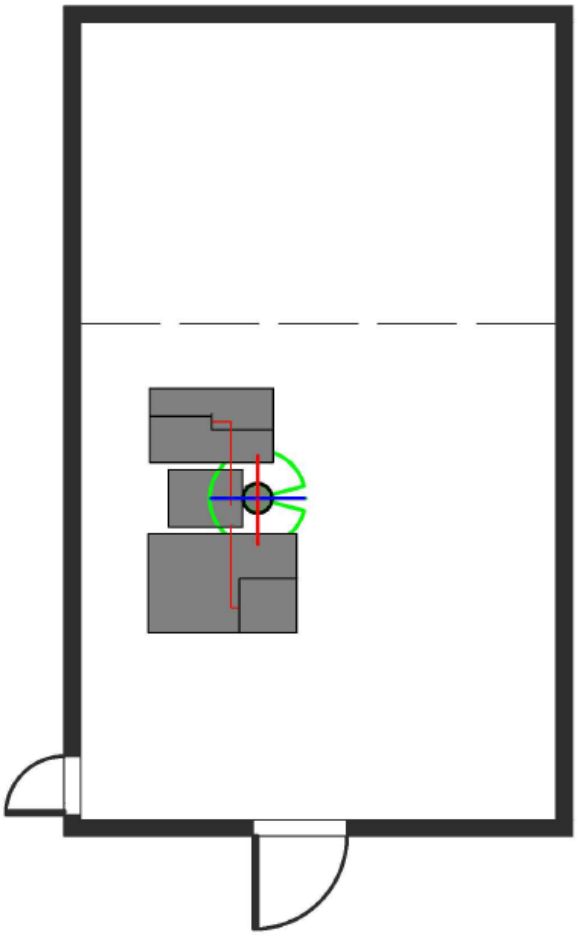


Figura 8.3 – Distribución final en planta

En la vista en planta, de arriba hacia abajo en la ilustaración se distinguen los equipos:

1. Láser Spirit
2. Brazo Robot
3. Láser AVIA

Además en las ilustraciones se representa el laboratorio, el volumen de trabajo del brazo robot y los haces láser ya direccionados.

Para direccionar los haces láser se emplea:

- Haz láser AVIA: Un espejo instalado sobre su mesa en tal forma que se direcciona el haz ortogonalmente hacia el brazo robot.
- Haz láser Spirit: Un periscopio formado por el conjunto de dos espejos. Mediante dos reflexiones ortogonales del haz se consigue esquivar el obstáculo formado por el propio equipo y direccionarlo hacia el brazo.

Los haces de ambos equipos se unen a la altura del brazo robot. Esta zona está encerrada dentro del volumen de trabajo descrito por el eje de la muñeca del brazo, por tanto el sistema de traslación puede orientar y trasladar la pieza bajo los haces.

Las dimensiones principales son las siguientes (las posiciones están tomadas con respecto a la esquina interior de las paredes que tienen puerta en el plano del suelo):

- Laboratorio:
  - Dimensiones: 577x965x236cm
- Láser AVIA:
  - Posición: 820x2770mm
  - Dimensiones mesa: 1800x1200x910mm
  - Haz:
    - Posición de salida: 1930x2560mm
    - Altura respecto a suelo: 1260mm
  - **Posición espejo: 1830x2570x1260mm**
- Láser Spirit:
  - Posición: 840x4330mm
  - Dimensiones mesa: 1500x900x900mm
  - Haz:
    - Posición de salida:1580x4830mm
    - Altura respecto a suelo: 1040mm
    - Altura obstáculo respecto mesa: 180mm
  - Posición periscopio:
    - **Espejo 1: 1820x4830x1040mm**
    - **Espejo 2: 1820x4830x1256mm**
- Brazo Robot:
  - **Posicion (ref mesa): 1070x3550mm**
  - Dimensiones mesa: 900x700x840mm
  - Dimensiones cilindro soporte brazo:
    - Radio: 180mm
    - Alura: 840mm
  - Radio volumen trabajo muñeca: 580mm

8.2. Diagrama final de la red

En el diagrama de abajo se representa la red de comunicaciones escogida para el laboratorio:

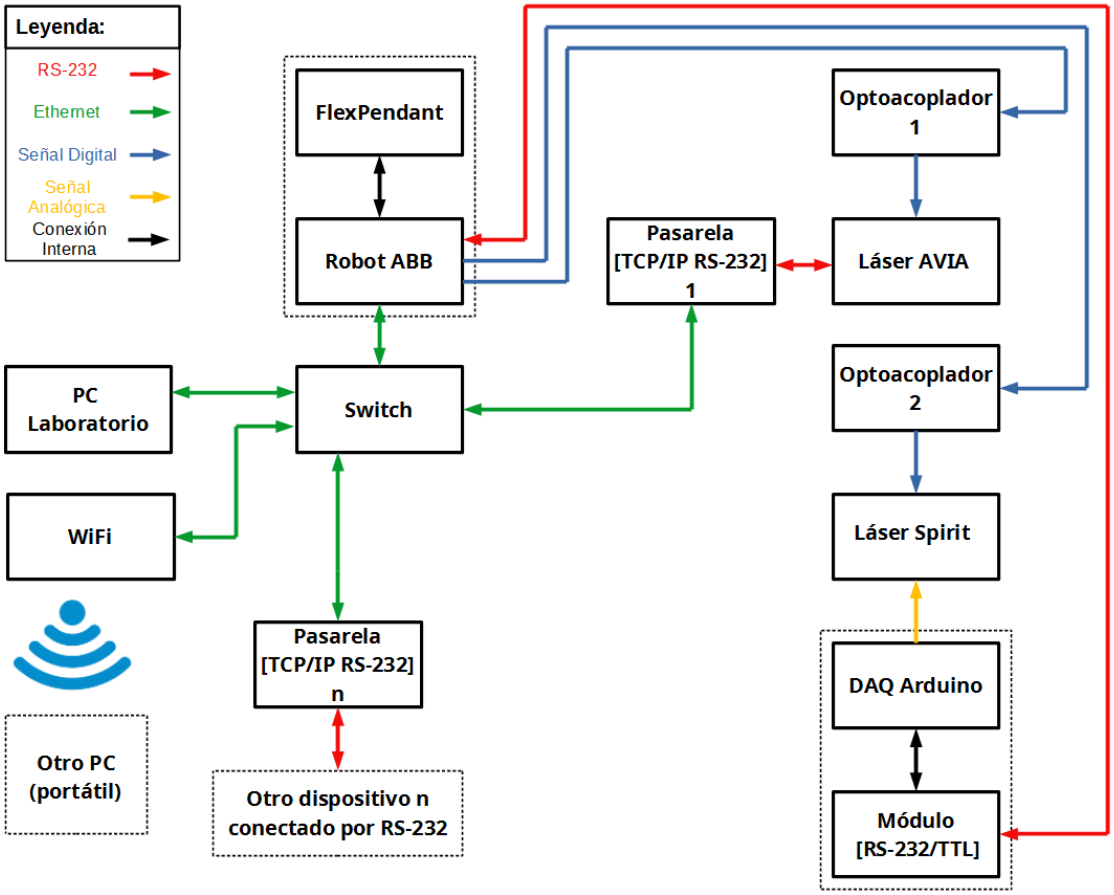


Figura 8.4 – Diagrama final de la red del laboratorio

A continuación se detallan las tareas y los enlaces entre elementos de la red en para cada uno de los mismos:

- **Configuración Láser AVIA:**
  PC/Portátil – Switch – Pasarela 1 – Láser AVIA
- **Programación Robot ABB:**
  PC/Portátil – Switch – Robot ABB
- **Disparo Láser AVIA:**
  Robot ABB – Optoacoplador 1 – Láser AVIA
- **Disparo Láser Spirit:**
  Robot ABB – Optoacoplador 2 – Láser Spirit
- **Modulación Láser AVIA:**
  Robot ABB – Switch – Pasarela 1 –Láser AVIA
- **Modulación Láser Spirit:**
  Robot ABB – [Módulo RS-232, DAQ Arduino] – Láser Spirit

- **Maniobra y Marcha/Paro:**
  Flex Pendant – Robot ABB - . . .

8.3. Sobre la conexión de los equipos

El siguiente listado comprende cuales son las distintas fuentes de alimentación de los distintos elementos de la red:

- Switch y wifi: Tienen fuentes de alimentación propias conectadas a las tomas monofásicas de 220V del laboratorio.
- Pasarelas WIZ750SR-110: Se han de alimentar con una fuentes de alimentación propias e independientes de 5V DC 1A conectadas a tomas monofásicas 220V del laboratorio.
- DAQ Arduino: Se puede alimentar con la toma de 24,5V DC que incorpora la controladora IRC5 Compact del brazo robot.
- Tarjeta salidas digitales controladora IRC5 Compact ABB (DSQC 652): Se ha de realizar conexionado externo para alimentar con la fuente de alimentación de 24,5V que dispone la controladora. Véase en la imagen el puente la tensión de alimentación y masa de la fuente (XS16) a la DSQC 652 (XS15).

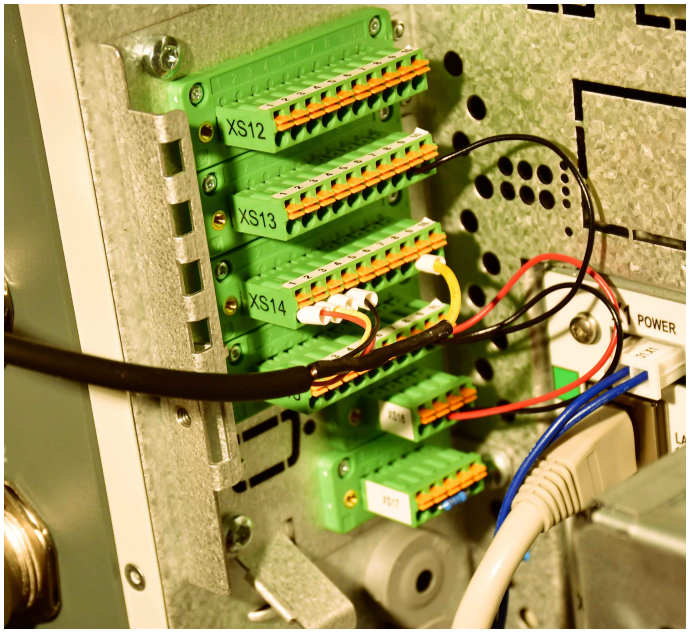


Figura 8.5 – Conexión tarjeta DSQC652

- Módulo optoacoplador disparo láser AVIA: No necesita alimentación.
- Módulo optoacoplador disparo láser Spirit: Se puede alimentar con la toma de 5V del conector SubD 21WA4 (User Control Interface) del própio láser Spirit.

En la siguiete imagen se puede ver la controladora del brazo robot (IRC5 Compact) con las conexiones en la parte central de:

- Salidas digitales para el disparo de los láseres (Cable negro de cuatro conductores).

- Puerto COM1 (RS-232) para conexión de la DAQ Arduino.

- Conexión ethernet.

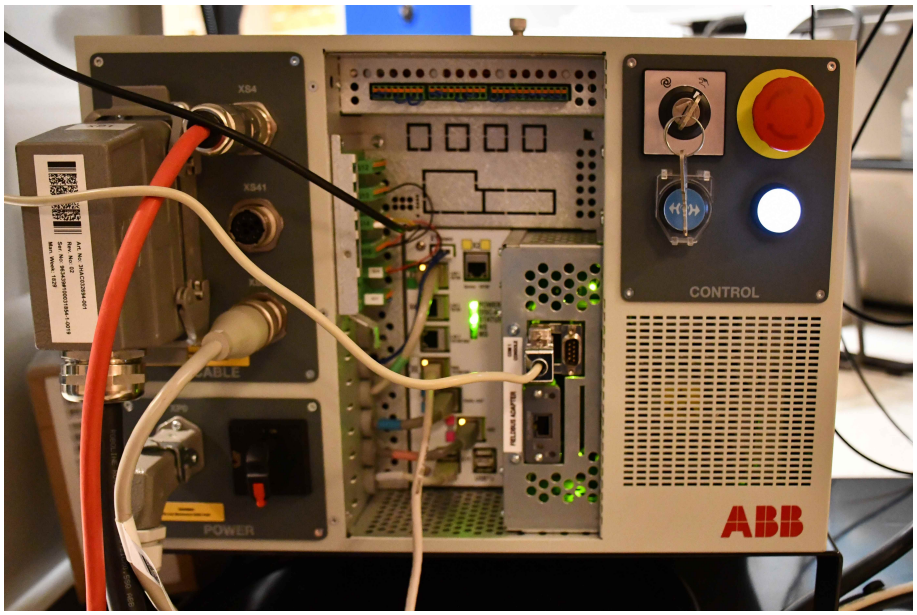


Figura 8.6 – Conexiones IRC5 Compact

8.4. Pasarelas WIZ750SR-110

Estas pasarelas se conectan actúan como interface entre el los equipos que comunican a través del estándar RS-232 y la red ethernet del Laboratorio. Para incorporar una nueva pasarela de este tipo o sustituir una ya incorporada así como para realizar cambios en la red se hace necesario cambiar la configuración de las mismas. Esto se hace a través del software Wizconfig proporcionado por su fabricante (Wiznet). Este software es muy ligero y no requiere instalación. Permite de manera sencilla configurar los parámetros de la comunicación serie y ethernet de la pasarela de manera sencilla. Para poder usar el software la pasarela debe estar en la mima red ethernet que el PC en el que se ejecuta la aplicación.

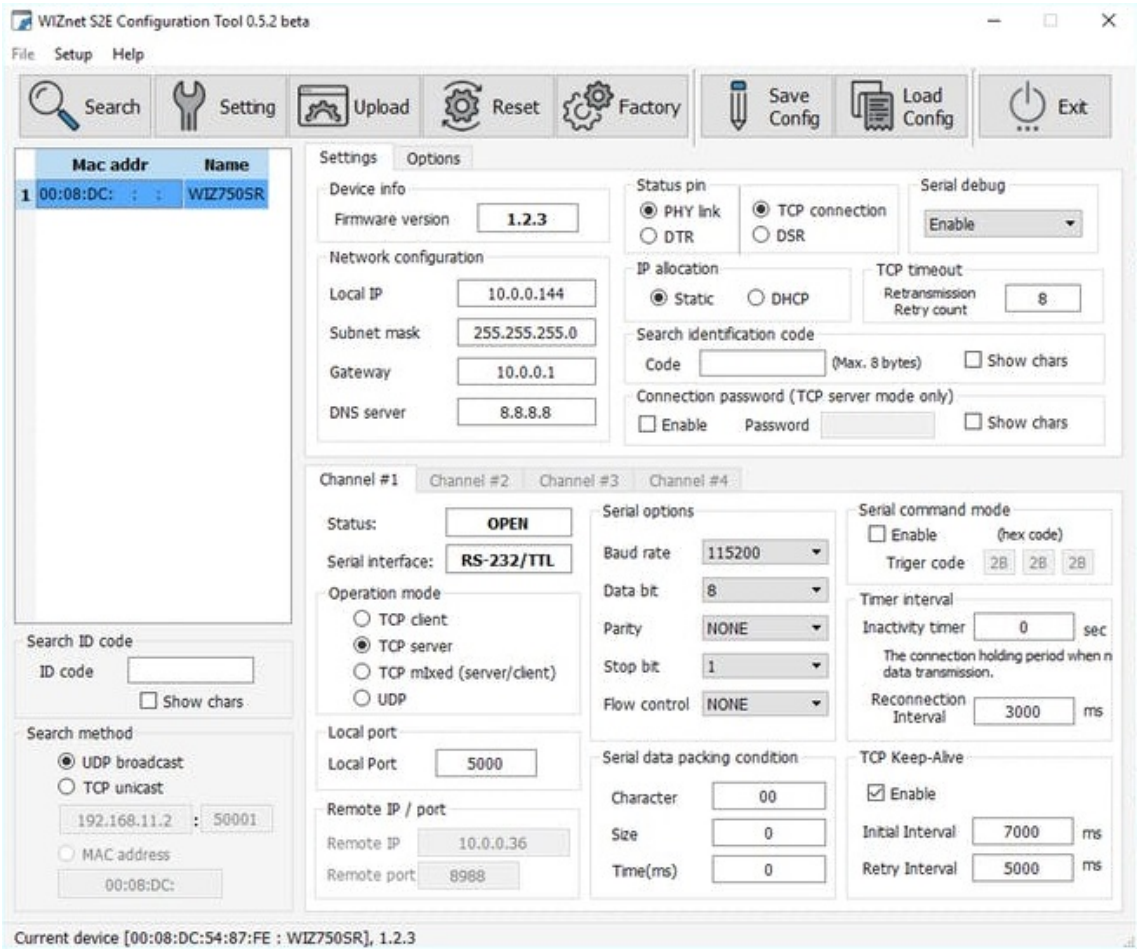


Figura 8.7 – Aplicación Wizconfig

Para la instalación de estas pasarelas en el laboratorio se optó por encapsularlas en una carcasa impresa en 3D y diseñada a tal efecto. La razón de no centralizar varias pasarelas u otros equipos en una misma carcasa es mantener la flexibilidad del laboratorio, ya que así es más sencillo hacer modificaciones.

Figura 8.8 – Carcasa WIZ750SR-110 1

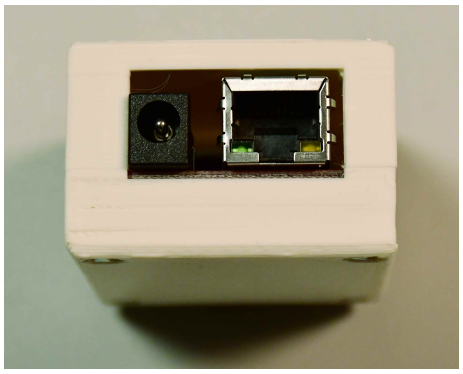


Figura 8.9 – Carcasa WIZ750SR-110 2

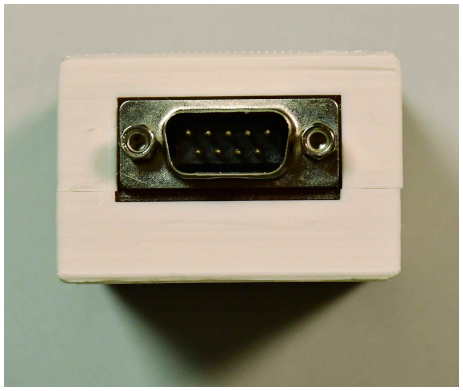


Figura 8.10 – Carcasa WIZ750SR-110 3

8.5. Solución disparo láseres

Con el principal objetivo de obtener una buena respuesta temporal del disparo se decidió emplear un canal de información independiente para este propósito. Mediante el uso de optoacopladores se consigue aislamiento galvánico adaptando a la vez los niveles de tensión de forma segura para los equipos ante un fallo.

El optoacoplador empleado es el integrado 4n35. Cuyas prestaciones en cuanto a sensibilidad son suficientes para optoacoplar estas salidas y no necesita alimentación auxiliar. (Habitualmente la alimentación auxiliar es empleada en estos circuitos integrados para aumentar la sensibilidad o trabajar con ciclos de histéresis mediante otra circuitería integrada).

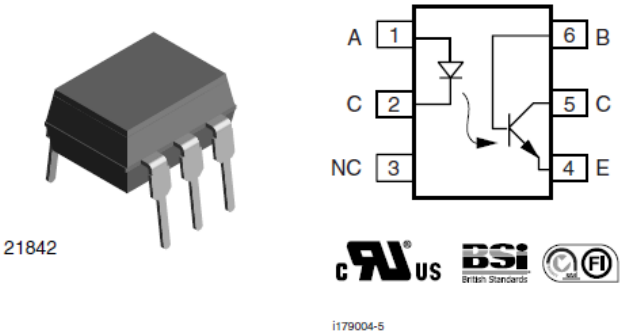


Figura 8.11 – Optoacoplador 4n35 DIP-6

Para el diseño de los circuitos se idealizó el comportamiento del optoacoplador como el del siguiente circuito equivalente:

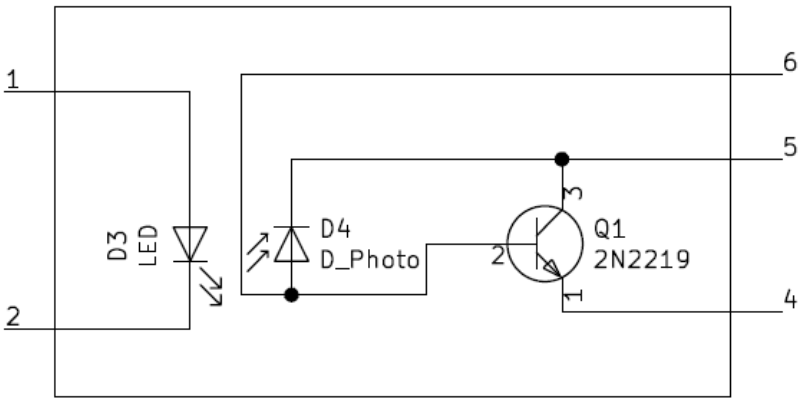


Figura 8.12 – Modelo ideal optoacoplador

En el circuito equivalente si hay diferencia de tensión entre los pines 5 y 4 y se polariza el LED, el transistor de la salida se polariza porque el fotodiodo conduce en inversa. Véase que aunque la base del transistor está accesible a través del pin 6, no se necesita.

Las soluciones aportadas constan cada una de un bloque al que entra la señal digital generada por el brazo robot y sale una señal adecuada para disparar cada uno de los equipos láser.

La solución es distinta para cada uno de los láseres:

- **Láser AVIA:**

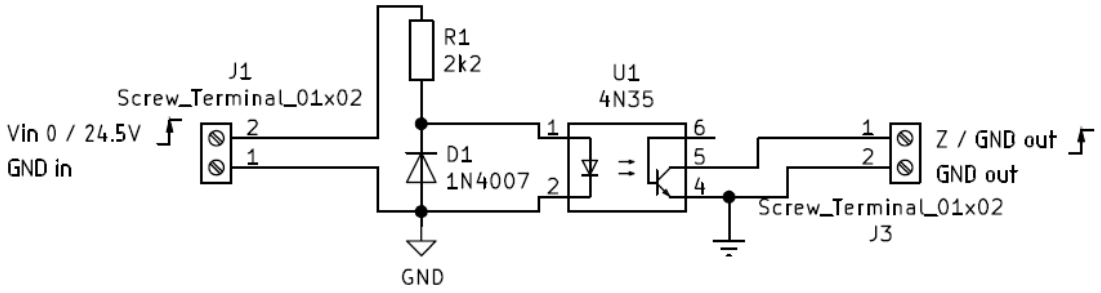


Figura 8.13 – Circuito optoacoplador láser AVIA

La señal de entrada proveniente del brazo robot conmuta entre 0 y 24,5V. Cuando se encuentra en 24,5V se polariza el diodo LED interno al optoacoplador U1 conmutando la salida del mismo. La resistencia R1 sirve para que se produzca en ella la caida de tensión adicional a la tensión de polarización del optoacoplador. El diodo D1 proteje el optoacoplador por si este se polarizara al revés.

En cuanto a la salida, en función de si la entrada del optoacoplador se encuentra polarizada o no esta conmuta o presenta alta impedancia respectivamente.

En la documentación del fabricante del láser se indica que se ha cerrar un contacto para disparar el equipo, pero no se especifica la circuitería interna al equipo. Dada esta circunstancia se supuso que esta circunstancia sería una resistencia de pull up o pull down.

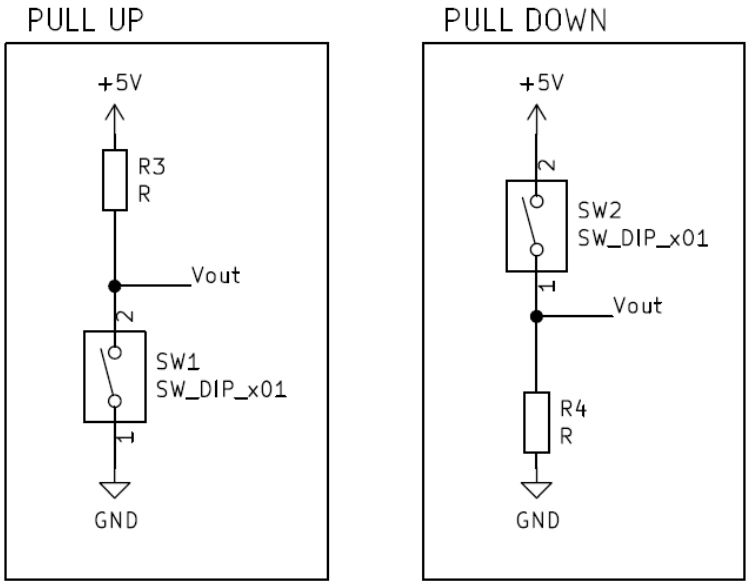


Figura 8.14 – Resistencias de pull up y pull down

En los esquemas de arriba están representados los circitos de pull up y pull down. En el caso del de pull up si el contacto está abierto a la salida se miden 5V y si se cierra 0V. En

el caso pull down el funcionamiento es inverso 0V abierto y 5 V cerrado. Cabe destacar que para un correcto funcionamiento el consumo de corriente a través de Vout ha de ser mínimo.

Se comprobó empiricamente que el comportamiento del equipo era el esperado y en base a eso se diseñó el circuito.

A la hora de diseñar el circuito no tiene importancia si la circuitería interna al láser es pull up o pull down en tanto a que el único factor limitante es que la corriente de polarización a través del fotodiodo depende del valor de la resistencia y la tensión y es independiente del tipo de montaje.

- **Láser Spirit:**

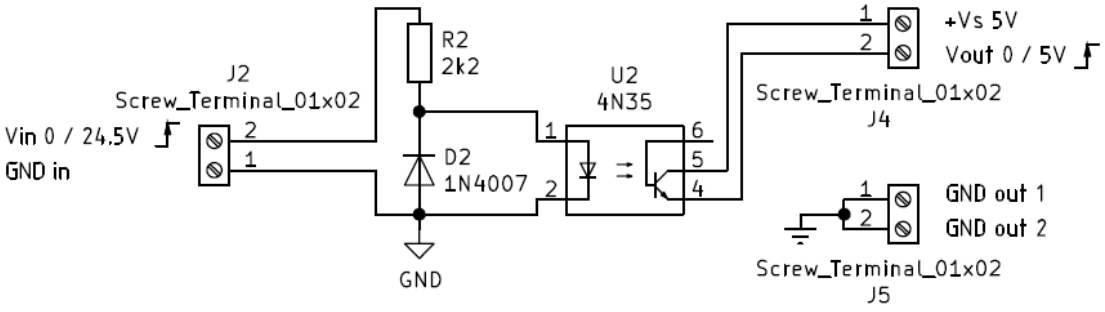


Figura 8.15 – Circuito optoacoplador láser Spirit

La entrada del optoacoplador es similar a la del caso del láser AVIA.

En cuanto a la salida, en este caso el fabricante tampoco especifica nada de la manera de disparar el equipo, más que ha de haber 5V en un contacto para dispararlo.

Por eso mismo en un primer momento se pensó en un montaje pull down con resistencia externa al equipo, ya que en este caso la alimentación también debía ser externa. Tras probar este montaje con varias resistencias se comprobó que no funcionaba, ya que no se consegía superar el umbral de tensión para disparar el equipo. Por tanto se supuso que la el equipo debía tener una resistencia interna. Se corrigió el montaje sin ninguna resistencia y se verificó que de este modo sí funcionaba.

En la salida del circuito hay 4 pines, dos son una masa común. Para conectar de salida se conecta una fuente de tensión de 5V DC entre el pin superior del esquema, los dos pines restantes son para la señal de disparo y su correspondiente masa.

De manera análoga a lo realizado con las pasarelas WIZ750SR-110 y por los mismos motivos de flexibilidad se obtó por instalar los optoacopladores en el interior de una pequeña caja impresa en 3D para cada uno:

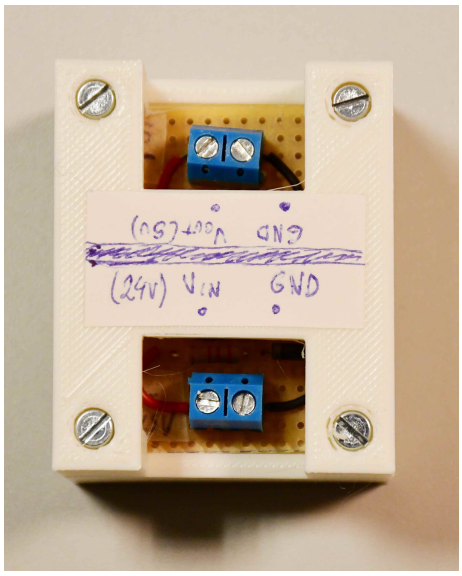


Figura 8.16 – Carcasa optoacoplador AVIA

Figura 8.17 – Carcasa optoacoplador Spirit

En las imágenes se puede observar que se disponen las bornas agrupadas en un lado o en otro del módulo. La razón de esto es indicar de forma intuitiva donde está el aislamiento galvánico y evitar posibles errores a la hora de realizar las conexiones.

8.6. Solución modulación de potencia en láseres

Para realizar el proceso de micromecanizado se necesita regular la potencia media del haz aplicada por los equipos láser.

Esto se puede hacer de dos maneras:

- Online: Consiste en ir variando los niveles de potencia a medida que se va mecanizado. En tal forma que el tratamiento puede ser más agressivo en las zonas que se precise.

- Offline: Consiste en establecer el nivel de potencia antes de comenzar el proceso y mantener este nivel constante a lo largo de todo el proceso.

Las soluciones planteadas permiten la regulación online ya que es parte de los requerimientos de este trabajo. La solución aportada es totalmente distinta para cada uno de los dos equipos láser.

8.6.1. Modulación de potencia en láser AVIA

Este equipo no dispone de una interface específica para este propósito, de modo que esta solución se vale de la interface de configuración de equipo para cambiar parámetros online. El parámetro que se cambia principalmente es la PRF, aunque la interface de configuración del equipo también permite variar la corriente en los diodos de bombeo energía.

Esta configuración online se realiza a través de la controladora del brazo robot que se conecta a través de ethernet al servidor TCP de la pasarela WIZ750SR-110 cuyo puerto serie RS-232 está conectado al láser AVIA.

8.6.2. Modulación de la potencia en el láser Spirit (DAQ Arduino)

La interface disponible en este equipo para la modulación de la potencia es una entrada analógica cuya tensión ha variar entre 0 y 5V.

Para realizar la modulación online de forma sencilla y con una dinámica temporal aceptable se ha de generar desde la controladora del brazo robot. Como esta controladora no dispone de una interface que permita generar la señal de salida optó por implementar una solución a medida. Esta solución a medida consiste en una tarjeta de adquisición (DAQ) basada en la placa Arduino UNO R3. La comunicación entre la DAQ y la controladora se realiza a través del puerto serie RS-232 disponible en la controladora.

A continuación se describe la DAQ Arduino en relación a diferentes aspectos.

8.6.2.1. Hardware DAQ Arduino

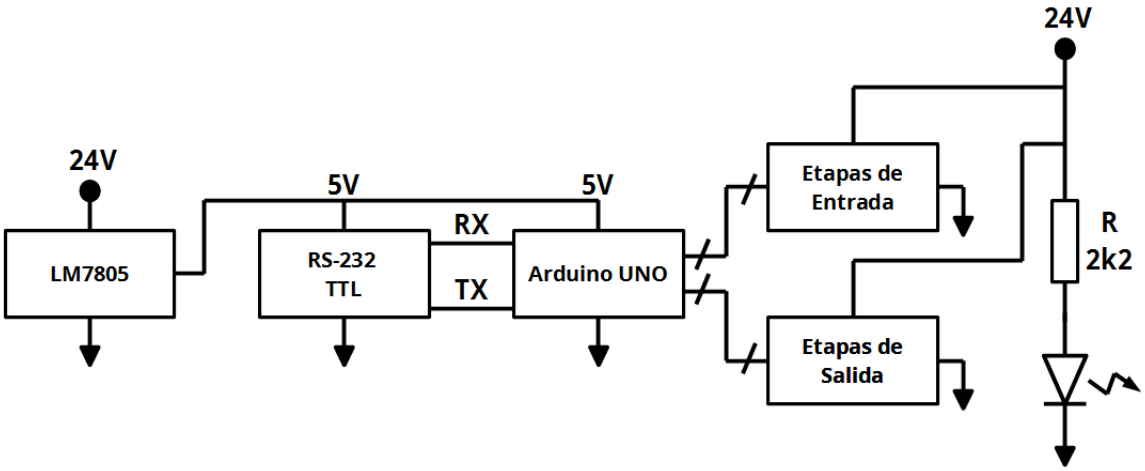


Figura 8.18 – Diagrama DAQ Arduino

El diagrama de arriba es una simplificación del esquema eléctrico de la DAQ. Se diferencian los elementos:

- LM7805: Este bloque engloba al regulador lineal LM7805 y la circuitería adicional para hacerlo funcionar correctamente. Su función es la de adaptar los 24,5V de alimentación de la DAQ a los 5V necesarios para alimentar el Arduino y el módulo RS-232 TTL. Requiere el montaje de un disipador y pasta térmica.
- RS-232 TTL: Representa un módulo que incluye el circuito integrado MAX232, el conector DB9 y la circuitería adicional para hacer funcionar el integrado. El integrado MAX232 actúa como conversor de niveles de las señales RXD y TXD de Arduino a los niveles del estándar RS-232. Se trata de un dispositivo que permite la comunicación bidireccional. Emplea condensadores para elevar el voltaje y permitir la conversión de niveles. El retardo temporal que produce la conversión no es significativo.

Figura 8.19 – Módulo RS-232 a TTL

- Arduino UNO: Este bloque se refiere a la placa Arduino UNO R3. Se trata de una placa se de software y hardware libre. Está basada en microcontrolador ATMega328p. Dispone de

6 entradas analógicas de 10 bits de resolución entre 0 y 5V a fondo de escala y 6 salidas analógicas PWM de 8 bits de resolución, los niveles de tensión de máximo y mínimo de la señal generada son 0 y 5V respectivamente, la frecuencia de esta señal se programó en el código a 62KHz.

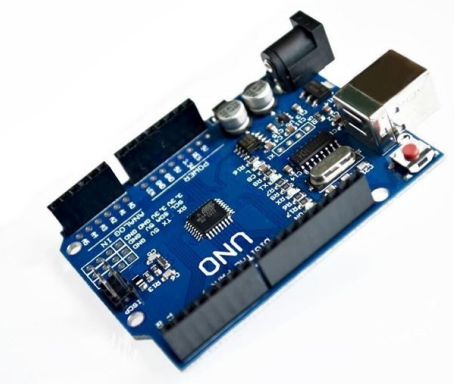


Figura 8.20 – Arduino UNO R3 (ATMega328p)

- Etapas de Entrada: Este bloque contiene la circuitería empleada para acondicionar las señales analógicas a medir.

  Los esquemas de la imagen son ejemplos representativos del acondicionamiento empleado en las entradas.

  El esquema superior es representativo de las entradas AI4 y AI5 de la placa. Consta de un seguidor de tensión y un divisor de tensión que permite adaptar niveles de tensión mayores de lo que puede convertir el convertidor AD del Arduino.

  En el esquema simplificado del conjunto el bloque de las entradas se alimenta entre 24,5V y masa. Eso es así porque entre estos niveles se alimentan los AO (Amplificadores Operacionales). Esta característica implica que ante una tensión negativa el AO satura a 0V y se protege el convertidor AD del Arduino.

  El esquema inferior de la imagen es representativo de las entradas AI0, AI1, AI2 y AI3 de la placa. El funcionamiento es similar al del circuito anterior pero en este caso sin el divisor de tensión.

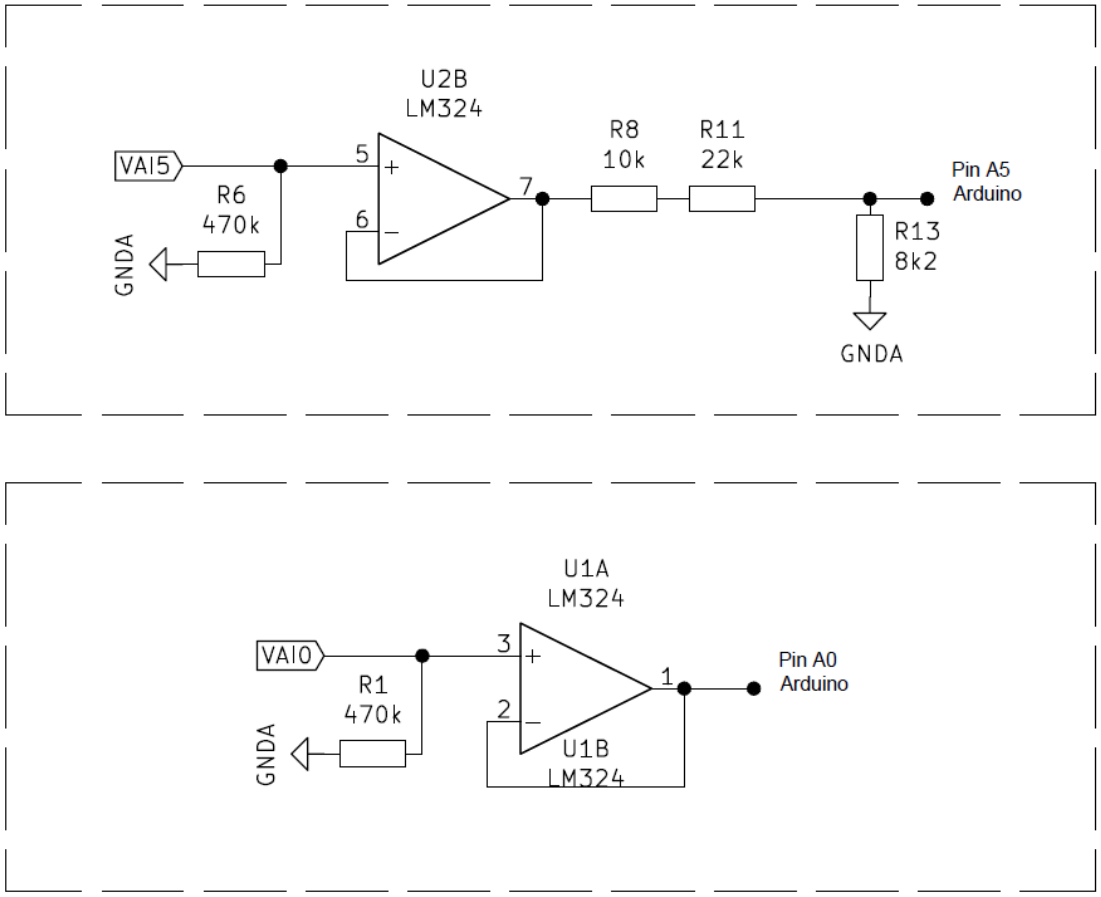


Figura 8.21 – DAQ Arduino circuito etapas de entrada

- Etapas de Salida: Este bloque contiene la circuitería empleada para acondicionar las señales analógicas generadas. Los esquemas de la imagen son ejemplos representativos del acondicionamiento empleado en las salidas.

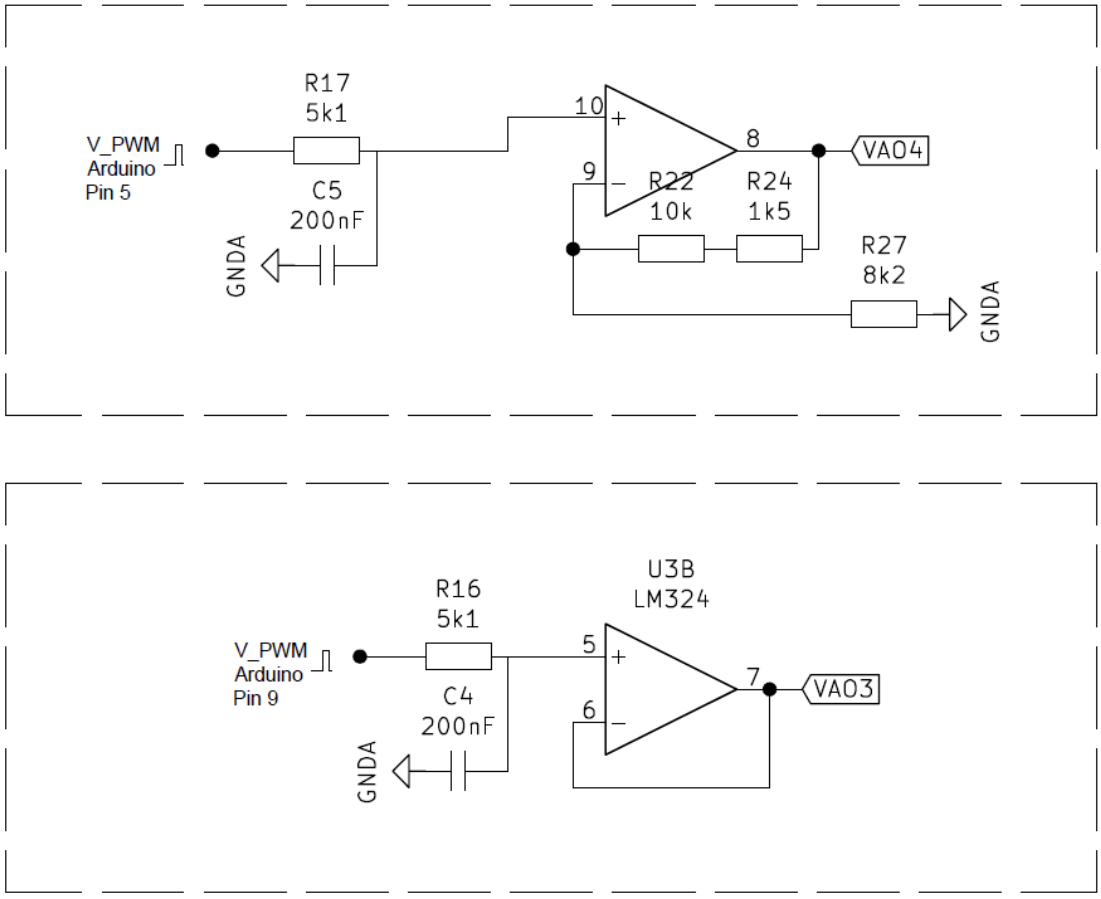


Figura 8.22 – DAQ Arduino circuito etapas de salida

La parte izquierda de ambos circuitos consta de un filtro paso bajo RC que sirve para convertir la señal PWM en una tensión contínua con un rizado mínimo. El retardo generado por estos filtros es del orden de 5ms.

En cuanto a la parte derecha de los circuitos, el de arriba es representativo de las señales AO4 y AO5 de la placa y consta de un AO en montaje no inversor con una ganancia adecuada en cada caso para adecuar los niveles de salida del filtro de 0 a 5V a los de cada salida, que son mayores. En la parte derecha del circuito inferior los AO se encuentran configurados como seguidores de tensión.

- Resistencia y LED: La función de estos componentes es que el led se encienda cuando la DAQ está alimentada.

Las siguientes tablas resumen las características de las entradas y salidas de la placa:

- Entradas (Resolución **10bits**):

Entrada	Fondo de escala
AI0	0-5V
AI1	0-5V
AI2	0-5V
AI3	0-5V
AI4	0-12V
AI5	0-24,5V (satura en 23V)

Tabla 8.1 – Canales de entrada DAQ Arduino

- Salidas (Resolución **8bits**):

Salida	Fondo de escala
AO0	0-5V
AO1	0-5V
AO2	0-5V
AO3	0-5V
AO4	0-12V
AO5	0-24,5V (satura en 23V)

Tabla 8.2 – Canales de salida DAQ Arduino

El montaje de todos estos elementos se llevo a cabo en una placa PCB diseñada a tal efecto. Todos los componentes de la placa son de tecnología de agujeros pasantes (THT) para una mayor facilidad de soldadura. Para los AO de las etapas de entrada y salida se emplearon tres circuitos integrados LM324-N DIP-14 (Cada uno de ellos contiene 4 AO).

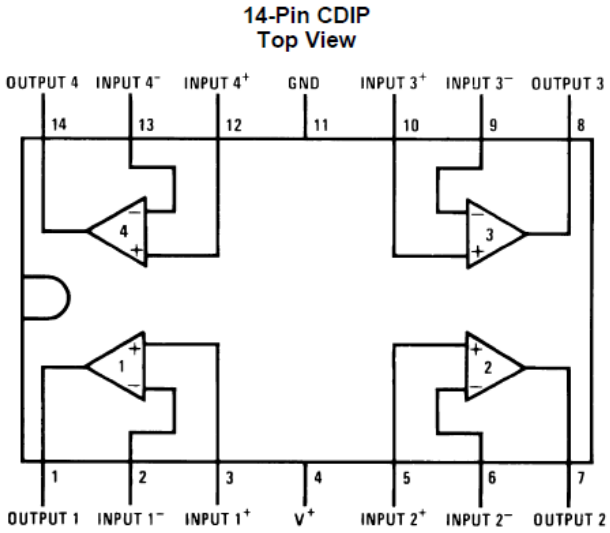


Figura 8.23 – Integrado AO LM324-N DIP-14

Otros aspectos a destacar del hardware son:

- Las entradas y las salidas son todas en modo común.

- Los LM324-N no se montaron directamente en la placa, sino a través de un socket, lo que permite la fácil sustitución en caso de que se dañen.

- Las conexiones con el Arduino se realizan mediante conectores Pin Header hembra, las entradas y salidas y alimentación a través de conectores Phoenix Terminal Block MKDS de dos bornas, el resto de uniones son soldadas.

- El consumo de corriente en las salidas se recomienda que sea inferior a 10mA.

- Todos los elementos se integraron en una carcasa modelada e impresa en 3D.

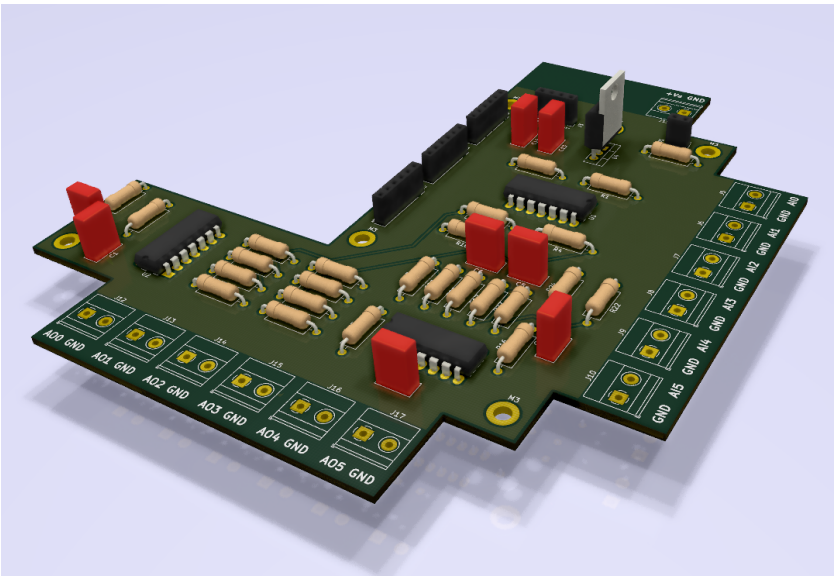


Figura 8.24 – Render Placa DAQ Arduino

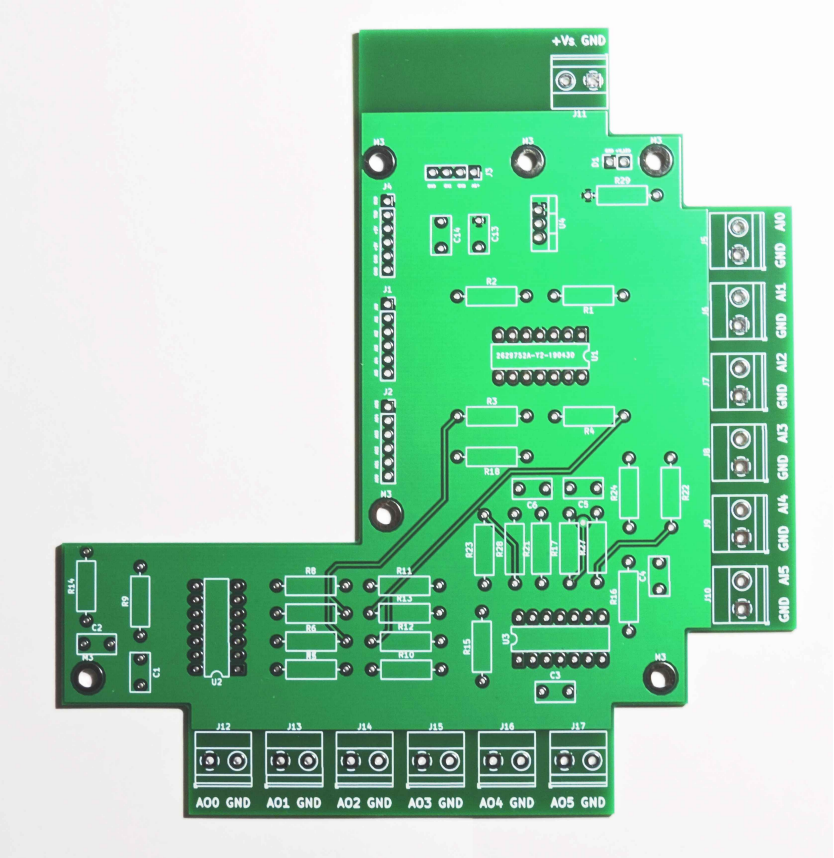


Figura 8.25 – Placa DAQ Arduino Front

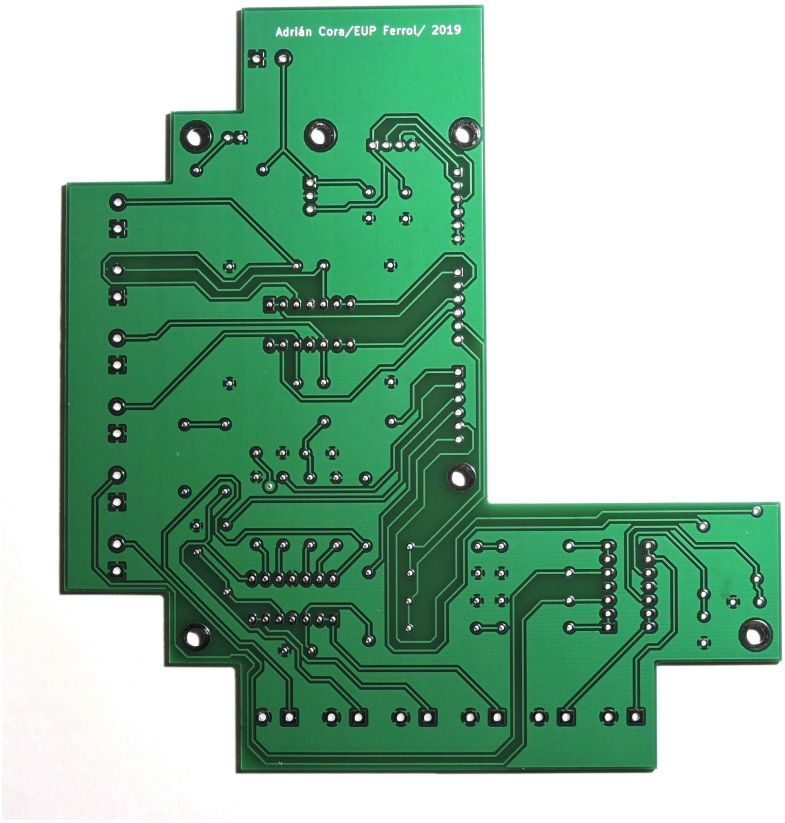


Figura 8.26 – Placa DAQ Arduino Bottom

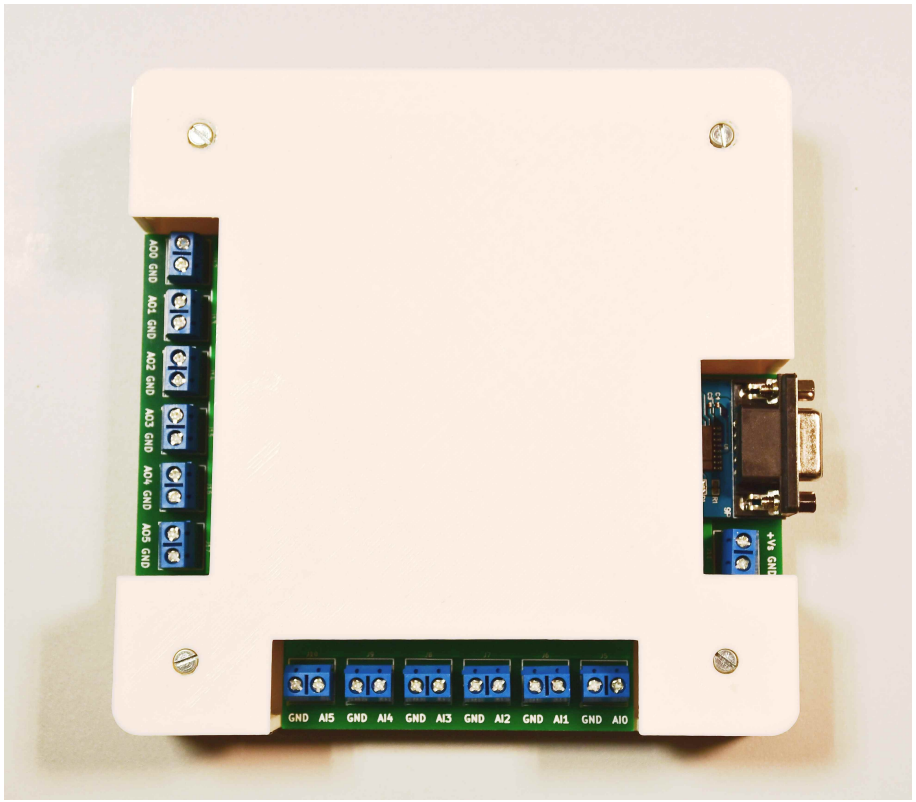


Figura 8.27 – DAQ Arduino 1

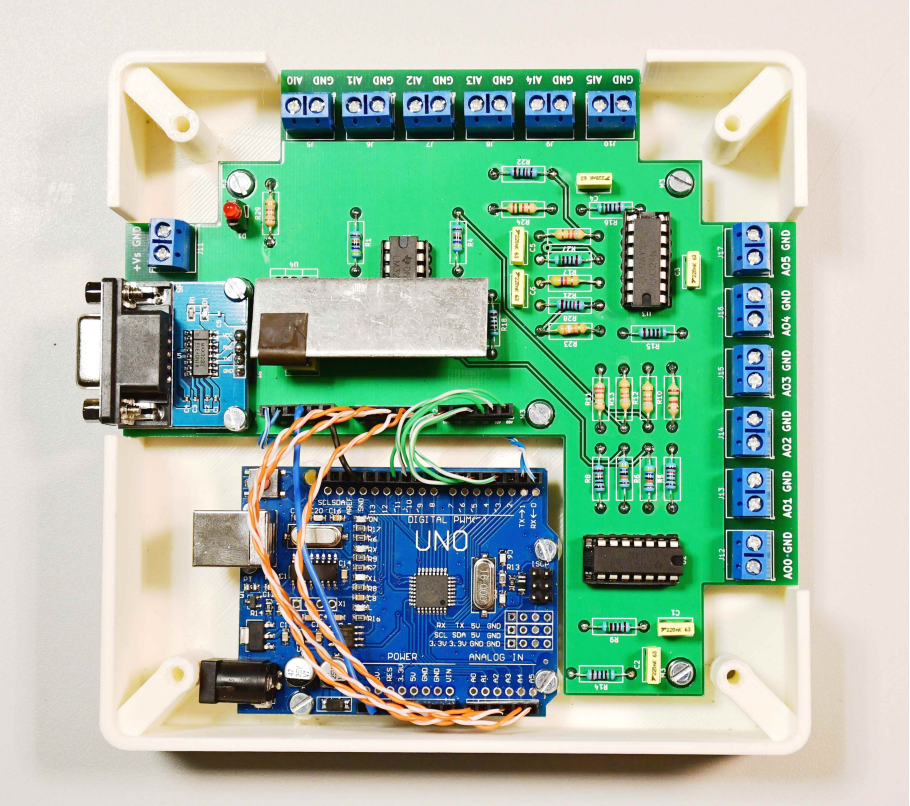


Figura 8.28 – DAQ Arduino 2

8.6.2.2. Software DAQ Arduino

El manejo de la DAQ se realiza mediante comandos a través del puerto serie RS-232. Se pueden enviar dos tipos de comandos, para la lectura de las entradas o para asignar valores a las salidas.

Cuando se envía un comando de forma correcta a la placa, esta contesta con otro comando. La función de la respuesta es, en caso de las salidas confirmar que se recibió el comando de forma correcta y se llevo a cabo la tarea asociada. En caso de las entradas, la respuesta a parte de contener la información de que entrada es leida, contiene el valor leido.

La velocidad tasa de envío de datos es de 9600 baudios por defecto, aunque se puede cambiar facilmente modificando el código fuente.

Las siguientes resumen los comandos posibles:

Símbolo	Significado
N	Valor numérico 0 a 255 para salidas 0 a 1023 para entradas
<CR>	Terminador Carácter retorno de carro ASCI 13
<LF>	Terminador Carácter nueva línea ASCI 10

Tabla 8.3 – Comandos DAQ Arduino 1

Salida	Comando	Respuesta
AO0	SPWM0<LF><CR>N<LF><CR>	SPWM0<LF><CR>N<LF><CR>
AO1	SPWM1<LF><CR>N<LF><CR>	SPWM1<LF><CR>N<LF><CR>
AO2	SPWM2<LF><CR>N<LF><CR>	SPWM2<LF><CR>N<LF><CR>
AO3	SPWM3<LF><CR>N<LF><CR>	SPWM3<LF><CR>N<LF><CR>
AO4	SPWM4<LF><CR>N<LF><CR>	SPWM4<LF><CR>N<LF><CR>
AO5	SPWM5<LF><CR>N<LF><CR>	SPWM5<LF><CR>N<LF><CR>

Tabla 8.4 – Comandos DAQ Arduino 2

Salida	Comando	Respuesta
AI0	SRAI0<LF><CR>	SRAI0<LF><CR>N<LF><CR>
AI1	SRAI1<LF><CR>	SRAI1<LF><CR>N<LF><CR>
AI2	SRAI2<LF><CR>	SRAI2<LF><CR>N<LF><CR>
AI3	SRAI3<LF><CR>	SRAI3<LF><CR>N<LF><CR>
AI4	SRAI4<LF><CR>	SRAI4<LF><CR>N<LF><CR>
AI5	SRAI5<LF><CR>	SRAI5<LF><CR>N<LF><CR>

Tabla 8.5 – Comandos DAQ Arduino 3

Los valores de tensión correspondientes a cada valor numérico de cada entrada o salida son:

- Entradas
  - AI0, AI1, AI2, AI3:
$$Vi = N\frac{5V}{1023} \tag{8.1}$$
  - AI4:
$$Vi = N\frac{12V}{1023} \tag{8.2}$$
  - AI5:
$$Vi = N\frac{24,5V}{1023} \tag{8.3}$$

- Salidas

  - AO0,AO1,AO2,AO3:

$$Vo = N\frac{5V}{255} \tag{8.4}$$

  - AO4:

$$Vo = N\frac{12V}{255} \tag{8.5}$$

  - AO5:

$$Vo = N\frac{24,5V}{255} \tag{8.6}$$

8.6.3. Soluciones para configuración, programación y monitorización de los equipos

En este apartado se describen las soluciones aportadas que posibilitan llevar a cabo todas las tareas del proceso que no tienen que ver con el disparo y la modulación de potencia en los láseres. Estas tareas son principalmente la configuración, programación y monitorización de los distintos equipos.

Desde el punto de vista de cada equipo, las distintas soluciones son:

8.6.3.1. PC del laboratorio y PCs portátiles

Se desarrolló una interface gráfica en Python. Esta interface consta de una ventana terminal que permite la comunicación manual con los distintos equipos a través de un puerto serie o a través de un servidor TCP. Permite configurar distintos parámetros de estas conexiones y los almacena en un fichero *.txt* para cargarlos de forma predeterminada cuando se vuelva a abrir la interface.

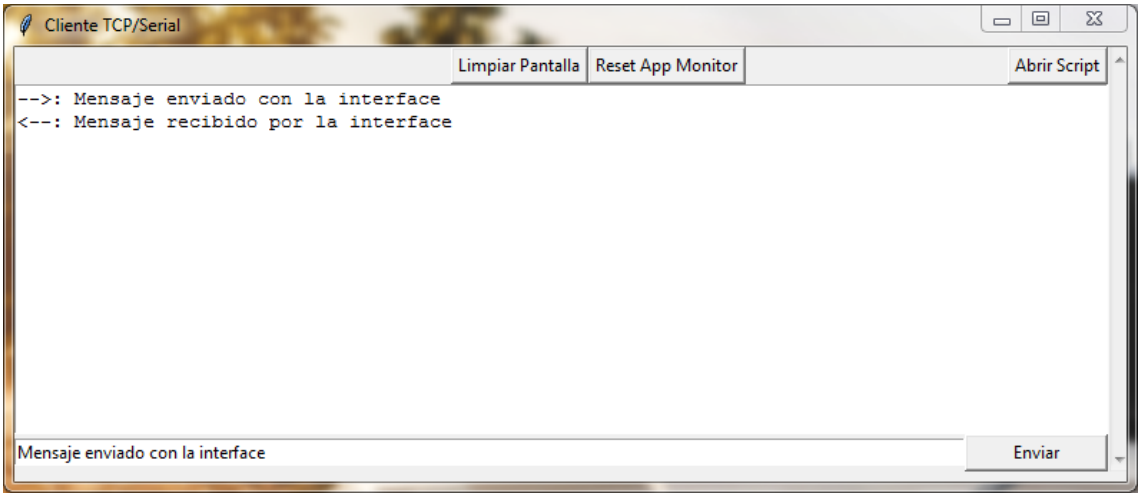


Figura 8.29 – Interface: Monitor

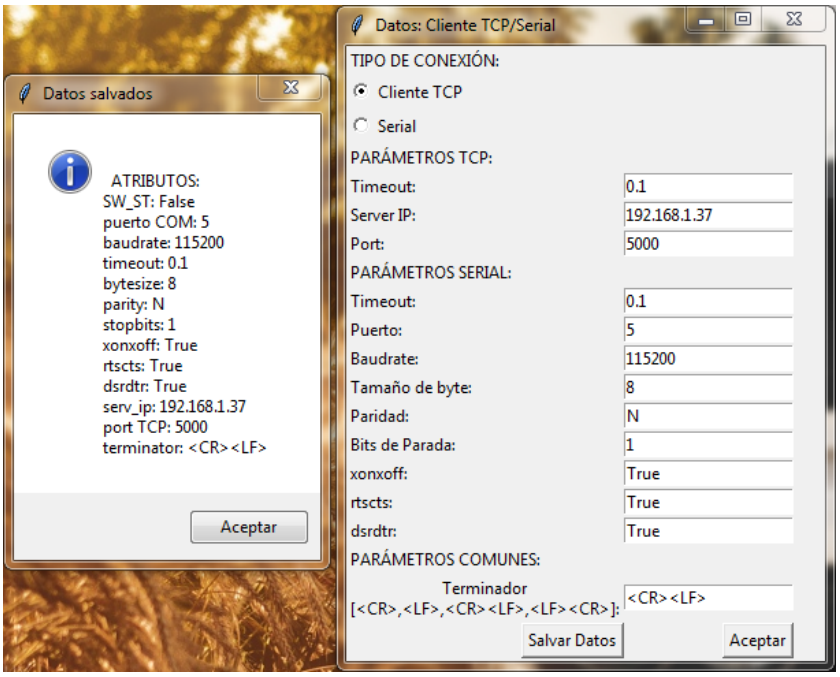


Figura 8.30 – Intrface: Parámetros

La principal utilidad de esta interface es realizar pruebas de conexión con los distintos equipos y realizar operaciones mediante comandos que no estén automatizadas.

Esta interface es un punto de partida para realizar tareas de configuración, programación y monitorización con los equipos. Su uso es de carácter general, es decir, no es específica para usar con ninguno de los equipos. Cabe destacar que es posible ejecutar varias de estas ventanas monitor en paralelo conectadas cada una a un equipo ejecutandolas en distinta ventanas de comandos de Python.

Para automatizar los conjuntos de tareas específicas particulares para cada equipo y configuración se propone el uso de scripts en Python.

La interface está programada en dos ficheros; *Interface.py* y *wizcom.py*. La razón de esto es que el primero de ellos contine principalmente el código correspondiente al manejo de la interface, mientras que el segundo contiene el código relativo a al manejo de las comunicaciones, de hecho, el segundo es importado en el código del primero.

De modo que la programación de scripts que permitan automatizar tareas rutinarias se puede hacer importando el módulo *wizcom.py* y empleando sus métodos, de este modo la programación de estos scripts resulta mucho más sencilla.

Desde el punto de vista de la programación en Python, el módulo *wizcom.py* define la clase Conexion. Esta clase tiene definidos los siguientes métodos:

- *__init__*: Se ejecuta cada vez que define un objeto de la clase. Crea en el objeto los atributos propios de la la comunicación:
  - *self.SW_ST*: Booleano que indica el tipo de comunicación True para serial, False para conexión a servidor TCP.
  - *self.port*: Parámetro comunicación serie. El puerto COM.

- *self.baudrate*: Parámetro comunicación serie.
- *self.timeout*: Parámetro comunicación serie.
- *self.bytesize*: Parámetro comunicación serie.
- *self.parity*: Parámetro comunicación serie.
- *self.stopbits*: Parámetro comunicación serie.
- *self.xonxoff*: Parámetro comunicación serie.
- *self.rtscts*: Parámetro comunicación serie.
- *self.dsrdtr*: Parámetro comunicación serie.
- *self.tcp_ip*: Parámetro comunicación a servidor TCP.
- *self.tcp_port*: Parámetro comunicación a servidor TCP.
- *self.terminator*: Parámetro común a ambas comunicaciones. Carácter ASCI que se añade al final de cada comando enviado. Admite uno o dos de los caractéres ASCI 13 y 10, que se han de poner como <CR> y <LF> respectivamente.

- *__str__*: Método para emplear un objeto de la clase con la función str(objeto) aporta una cadena de caracteres con la información de todos los atributos de la clase.
- *open*: Abre el canal de comunicación definido por el parámetro *self.SW_ST* de la clase.
- *close*: Cierra el canal de comunicación definido por el parámetro *self.SW_ST* de la clase, en caso de que esté abierto.
- *write*: Escribe un comando añadiendo el termiandor correspondiente a lo definido en los atributos de la clase.
- *read*: Lee los datos del buffer de entrada, devuelve una cadena nula cuando está vacío.
- *aplicaconfiguracion*: Aplica a un objeto de la clase la configuración guardada en un fichero .txt. Por defecto Configuración.txt aunque admite cualquier nombre como parámetro de entrada.
- *salvaconfiguracion*: Guarda la configuración actual en un fichero .txt de un objeto de la clase. Por defecto el fichero Configuración.txt aunque admite cualquier nombre como parámetro de entrada.

8.6.3.2. Láser AVIA

Este equipo permite la configuración y consulta de sus parámetros a través del puerto serie RS-232. También incorpora de forma nativa un display LCD y una serie de botones para la configuración manual.

Se propone la configuración y monitorización de parámetros en este equipo mediante la interface en Python creada a tal efecto, así como, de una manera mas automatizada a través de scripts en Python que hagan uso del módulo *wizcom.py*.

Véase que empleando el código creado en Python el método de conexión al equipo es muy flexible, ya que permite de manera sencilla alternar entre una conexión serie RS-232 entre el PC y el equipo o una conexión a traves de la pasarela WIZ750SR-110 empleando la conexión a un servidor TCP desde el PC.

En el manual del equipo aparecen tabulados todos los comandos y la manera en que se debe de configurar el equipo. Por eso mismo carece de interés extenderse en este punto más allá de cómo establecer la comunicación.

8.6.3.3. Láser Spirit

Este equipo no dispone de las mismas posibilidades de configuración y monitorización que el láser AVIA. El equipo dispone de un PC portatil propio (proporcionado por el fabricante) que corre un software cerrado. Desde este PC se pueden llevar a cabo las tareas de monitorización y configuración.

8.6.3.4. Brazo robot

Como se explica en otros apartados el que es referenciado como brazo robot se compone de dos equipos de ABB; brazo robot IRB 120 y controladora IRC5 Compact la cual dispone de la HMI (Interface Hombre Máquina) FlexPendant. A su vez la controladora IRC5 Compact es como un PC conectado en red que corre el sistema operativo RobotWare y dispone de un diso duro.

Conectandose a la red es posible acceder al almacenamiento en el disco duro de la controladora pudiendose de este modo introducir o eliminar programas en RAPID, que luego se pueden ejecutar desde el FlexPendant. De este modo es sencillo programar y operar el equipo.

Lo que a continuación se describe es una serie de módulos programados en RAPID cuyo fin es el de facilitar el manejo desde la controladora del disparo y la modulación de potencia en los equipos láser. Importando estos módulos en los códigos de RAPID generales del proceso de micromecanizado es mucho más sencillo e inteligible programar la realización de estas tareas.

- **Disparo:** Para realizar el disparo de los láseres hay que cambiar el estado de la salida digital correspondiente de la controladora. Esto consiste en la modificación de una variable booleana reservada a tal efecto. Dada la sencillez de esto en la programación se entiende que no es necesario escribir un módulo a tal efecto. Sin embargo cabe destacar que el sistema operativo integra de forma nativa las variables de las salidas digitales y permite asignarles otros nombres. Los nombres por defecto tienen la forma sdn donde n es el número correspondiente al de la salida en concreto.

- **Modulación de potencia en láser AVIA (AviaTcp.mod):** La modulación de potencia en este equipo se realiza a través de la conexión a su pasarela correspondiente WIZ750SR-110, que actúa como si fuese un servidor TCP. Por tanto el código en RAPID correspondiente facilita la conexión envío y recepción de datos a un servidor TCP.

- **Modulación de potencia en láser Spirit (SpiritArduino.mod):** La modulación de potencia en este equipo se realiza a través de una entrada analógica. Como la controladora no disponía de salidas analógicas se implementó una DAQ basada en Arduino conectada a la controladora a través del puerto serie que permite generar estas salidas. Por tanto el código en RAPID correspondiente consiste en la apertura del puerto serie RS-232 COM1 de la controladora y el envío y lectura de comandos a través del mismo.

8.7. Pruebas de funcionamiento

Se realizaron una serie de pruebas para comprobar el desempeño del conjunto. Estas pruebas están orientadas a evaluar el desempeño durante el proceso de micromecanizado con los distintos equipos láser.

Se realizaron medidas con osciliscopio de las tensiones más relevantes en los elementos dimensionados y se llevó a cabo la inspección visual de una serie de muestras sometidas al proceso de micromecanizado. Por último, cabe destacar que también se tuvo en cuenta el desempeño del conjunto desde un punto de vista global.

El osciloscopio con el que se realizaron las mediciones disponía de una interface RS-232 a través de la cual fue sencillo guardar las medidas realizadas en un fichero, que después se procesó con MATLAB.

8.7.1. Micromecanizado con el láser AVIA

La magnitud medida en este equipo fue la tensión en la entrada de disparo del láser, o lo que es lo mismo, la tensión en la salida del optoacoplador.

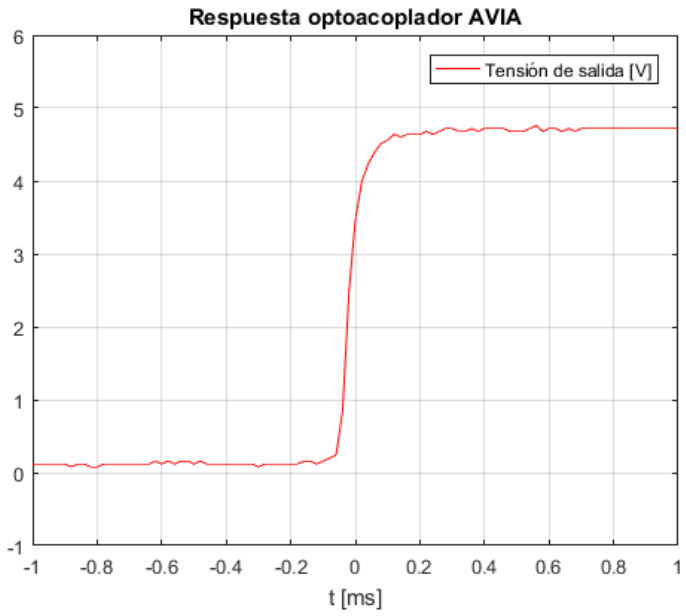


Figura 8.31 – Optoacoplador AVIA subida

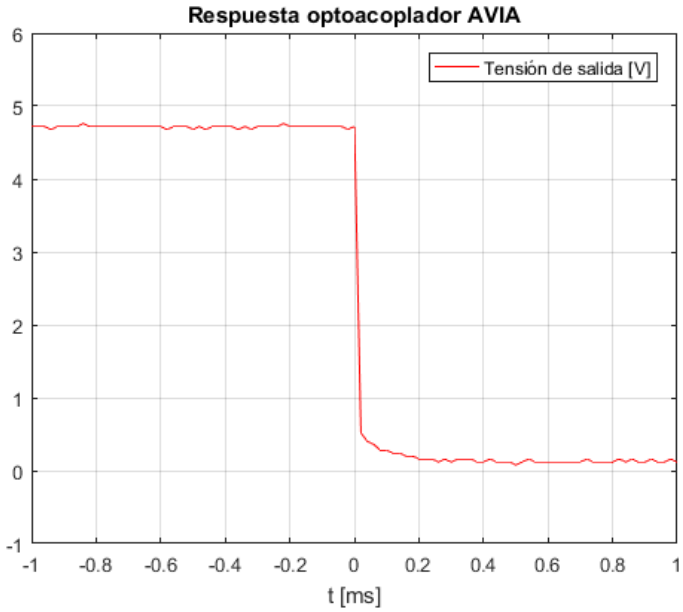


Figura 8.32 – Optoacoplador AVIA bajada

Las siguientes imágenes corresponden al resultado del micromecanizado sobre un recorte de papel de oficina convencional (DINA4) y sobre un recorte de acetato blanco. En el caso de la segunda muestra cada linea se hace a una PRF determinada, según se asciende en esta imagnen a cada linea aumenta este valor quedando demostrado que la modulación de potencia a través del protocolo TCP y variando la frecuencia funciona de forma correcta.

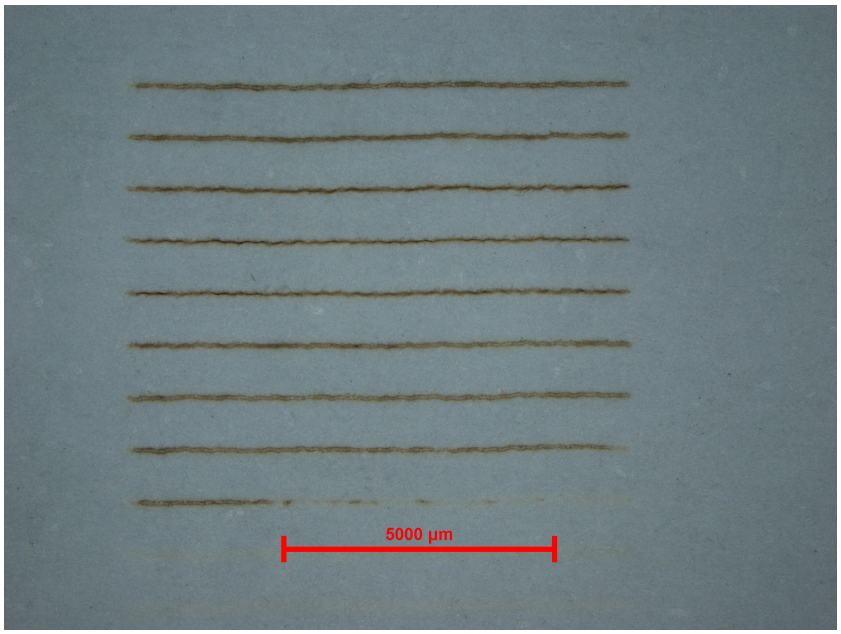


Figura 8.33 – Mecanizado AVIA sobre papel

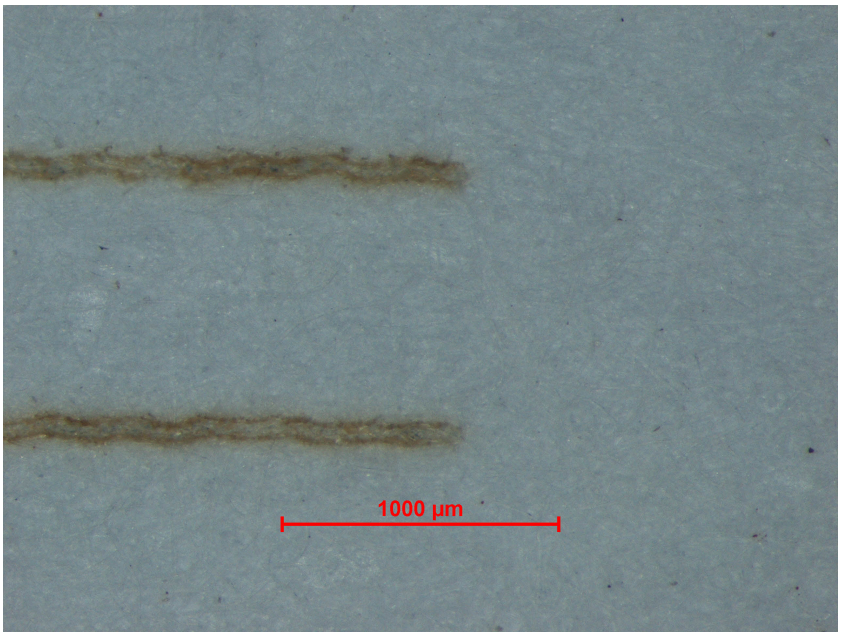


Figura 8.34 – Mecanizado AVIA sobre papel zoom

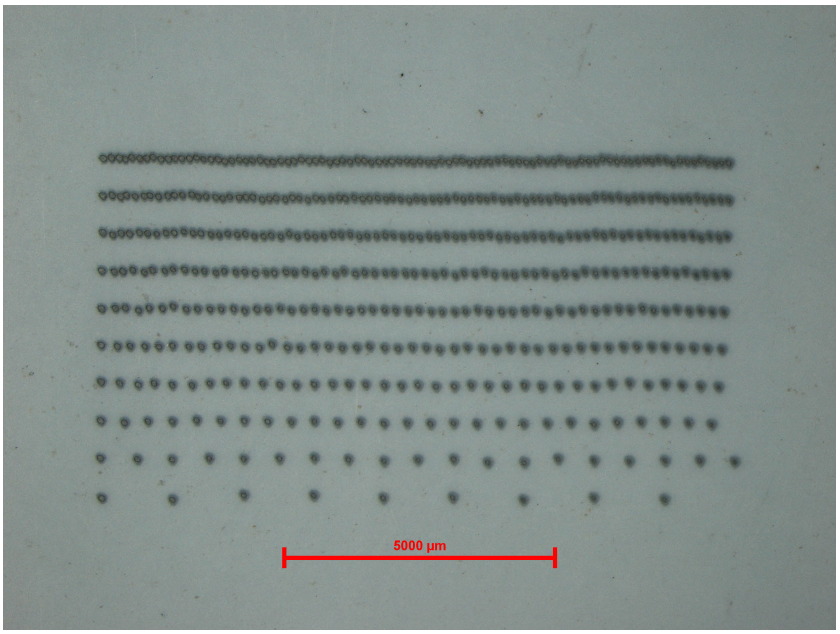


Figura 8.35 – Mecanizado AVIA sobre acetato

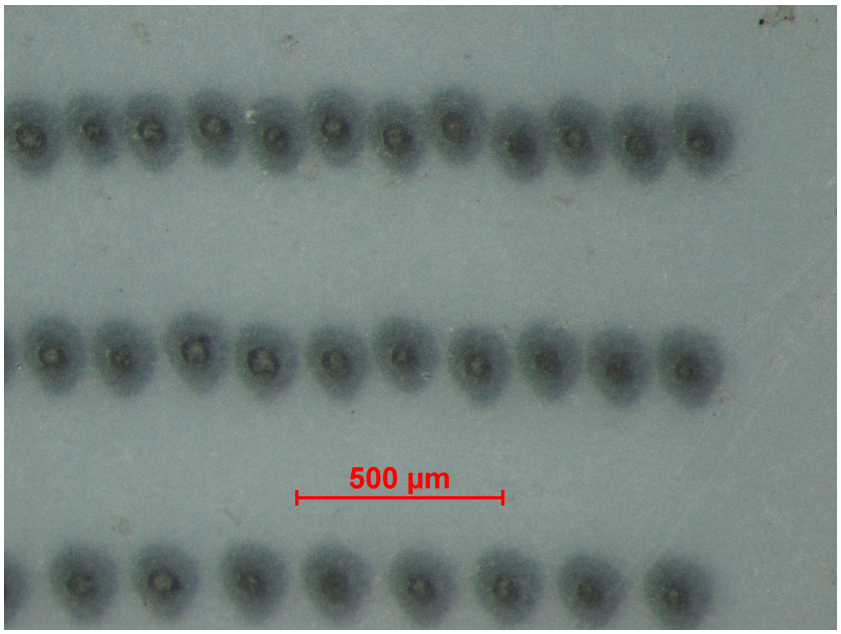


Figura 8.36 – Mecanizado AVIA sobre acetato zoom

Desde el punto de vista global del desempeño, la coordinación de la modulación de potencia a través de TCP y el disparo a tavés del optoacoplador se realiza de forma correcta según lo esperado.

8.7.2. Micromecanizado con el láser Spirit

En el caso de este equipo se realizaron medidas con el osciloscopio tanto de la señal optoacoplada correspondiente a su disparo, como del desempeño de la DAQ Arduino implementada para la modulación de la potencia.

Como para modular la potencia se necesita únicamente una tensión de salida analógica de 0 a 5V, todas las medidas correspondientes a la DAQ Arduino fueron realizadas sobre el canal AO1.

- Gráficas del disparo

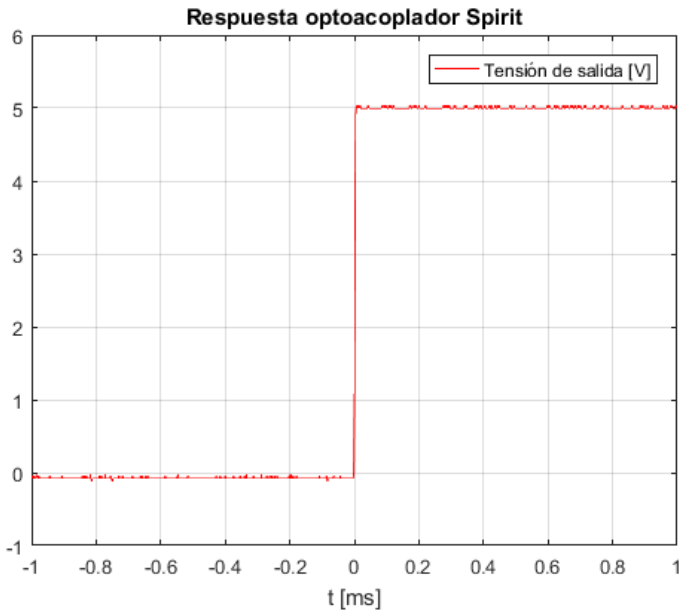


Figura 8.37 – Optoacoplador Spirit subida

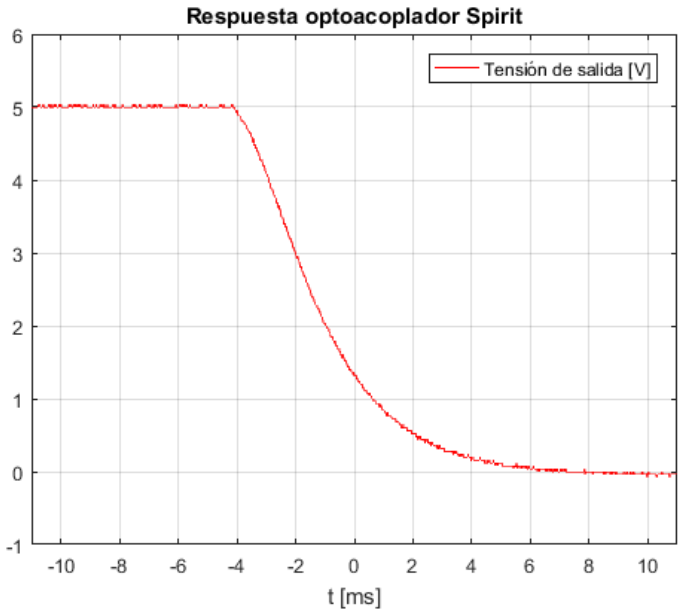


Figura 8.38 – Optoacoplador Spirit bajada

- Gráficas del desempeño de DAQ

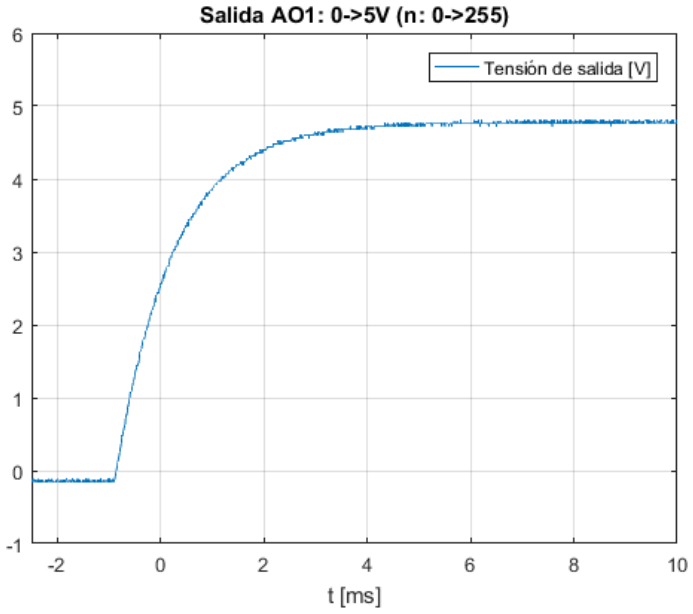


Figura 8.39 – Respuesta DAQ Arduino 0 a 5V

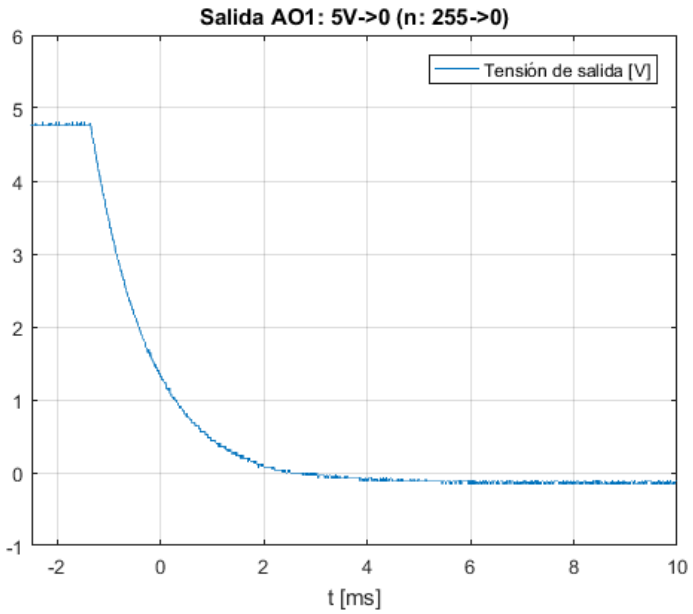


Figura 8.40 – Respuesta DAQ Arduino 5V a 0

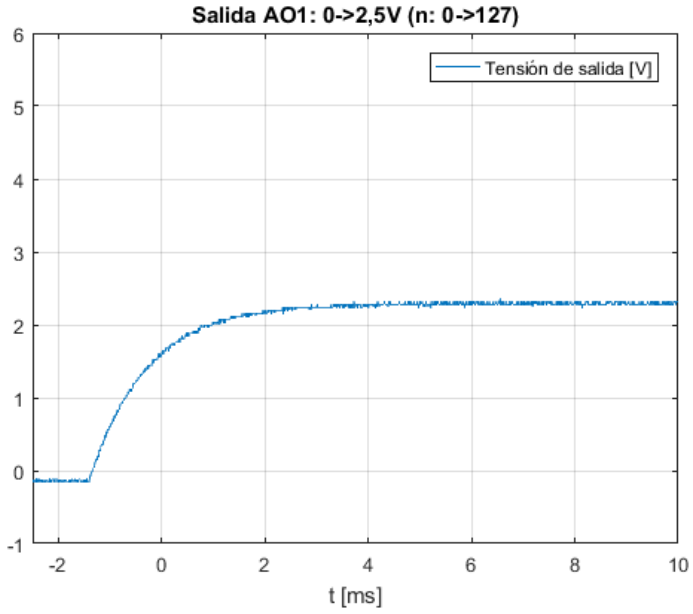


Figura 8.41 – Respuesta DAQ Arduino 0 a 2,5V

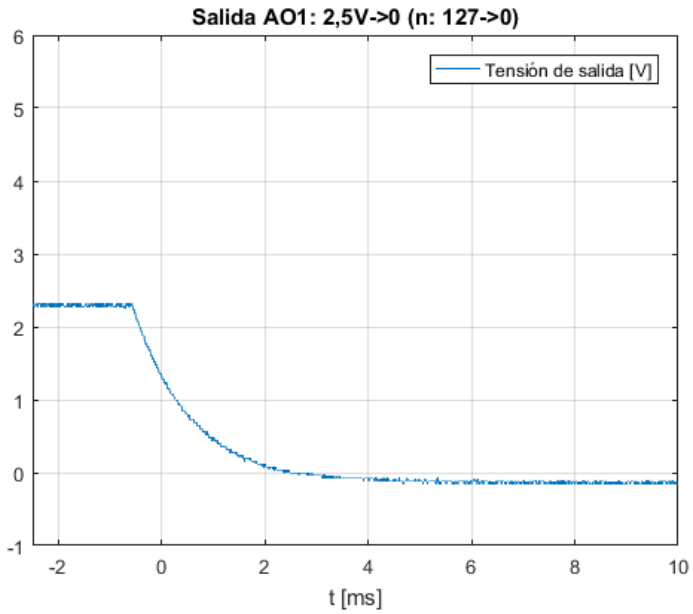


Figura 8.42 – Respuesta DAQ Arduino 2,5V a 0

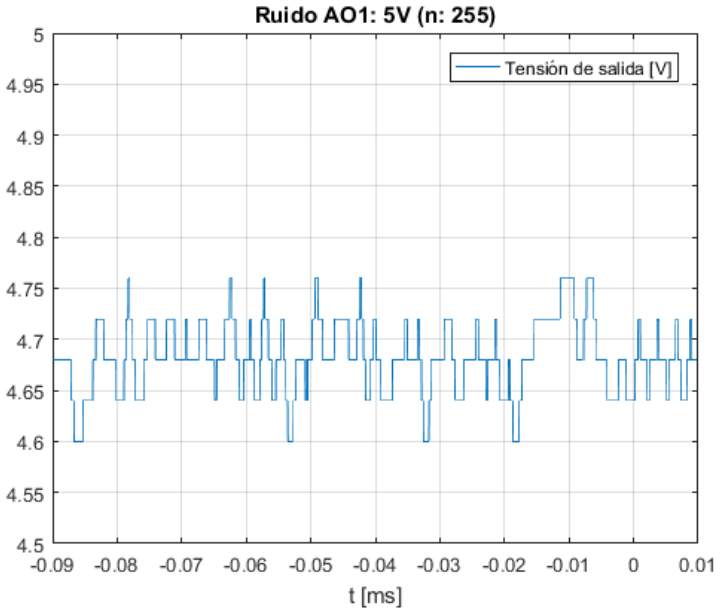


Figura 8.43 – Respuesta DAQ: detalle del ruido a 5V

- Gráficas del desempeño de DAQ

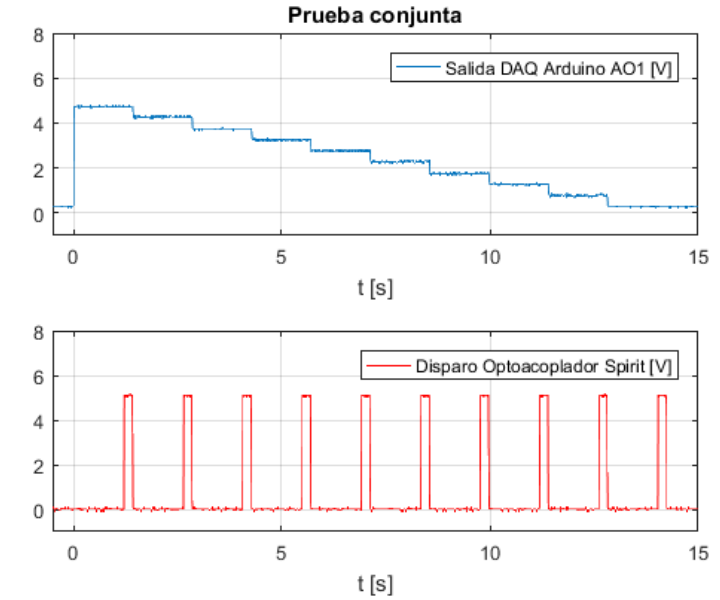


Figura 8.44 – Respuesta conjunta DAQ Arduino y Optoacoplador Spirit

Las siguientes imágenes corresponden al resultado del micromecanizado sobre un recorte de acetato blanco. En la primera imagen se puede ver que las lineas de arriba tienen un mayor grosor que las de abajo (y aunque ahí no se aprecie el mecanizado también es más profundo). Esto es así porque se a cada linea que se varía la potencia proporcionalmente.

En contraposición con la pruebas con el láser AVIA no se puede bajar la frecuencia para ver los distintos pulsados en la pieza. Esto se debe a que la emisión pulsada es del tipo mode locking en vez de Q-switched lo que implica que internamente la potencia del láser no se

modula variando la frecuencia de repetición de pulsos.

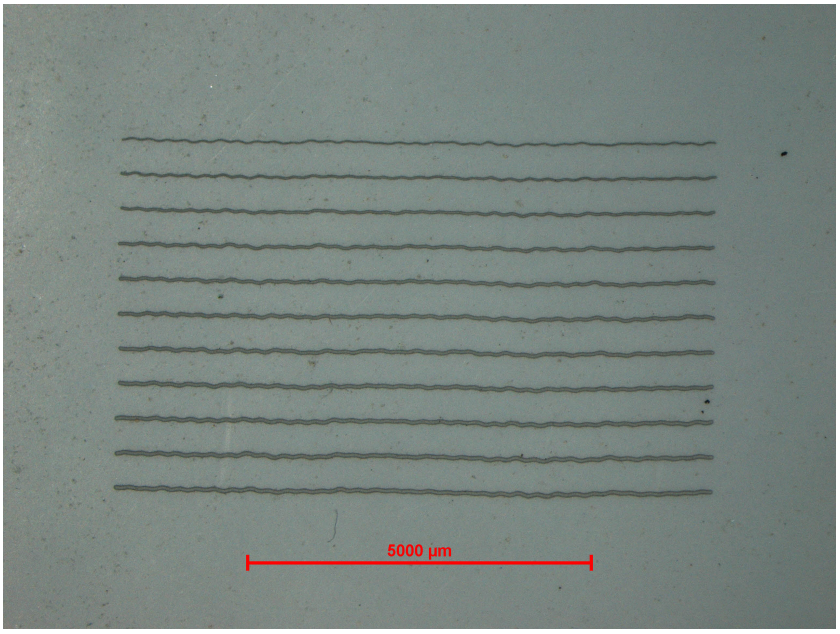


Figura 8.45 – Mecanizado Spirit sobre acetato

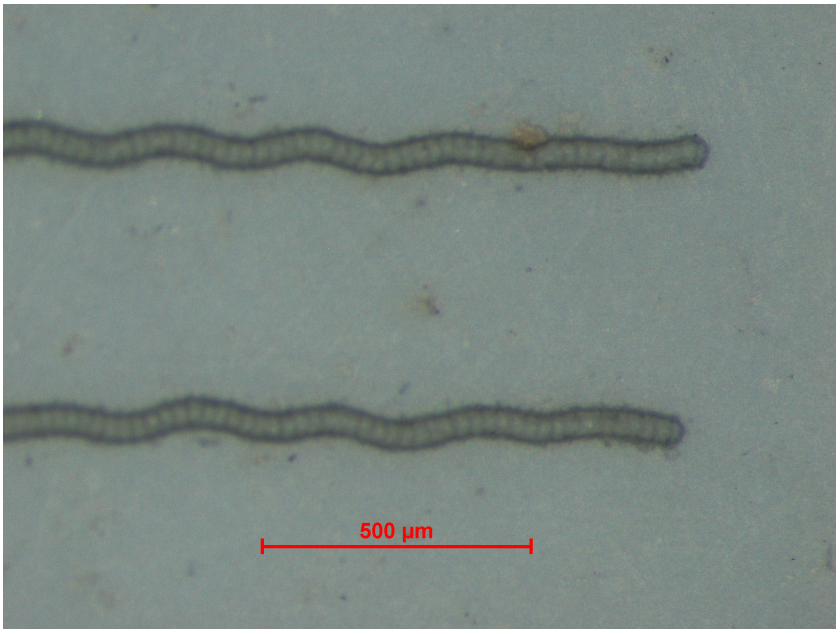


Figura 8.46 – Mecanizado Spirit sobre acetato zoom

8.8. Evaluación de los resultados

La evalución de los resultados se plantea en dos puntos. Desde el punto de las capacidades de la red de comunicaciones del laboratorio y desde el del proceso de micromecanizado.

Respecto a la red del laboratorio, esta aumentó sus capacidades:

- Se eliminó el cuello de botella que suponía la comunicación de los equipos a través de

RS-232 mediante la correcta instalación de las pasarelas WIZ750SR-110 empleando el protocolo TCP.

- Se ampliaron las capacidades del brazo robot con la tarjeta implementada DAQ Arduino.
- Se dispuso el uso de Python 3 como lenguaje de carácter general con un módulo que facilita la comunicación TCP y serie RS-232. Así como una interface en este lenguaje que consta de una ventana de comandos tipo monitor (muy útil para realizar pruebas y configuraciones no automatizadas mediante scripts a medida en los equipos).
- Se configuró el brazo robot ABB y se programaron una serie de módulos que permiten llevar a cabo de forma sencilla el proceso de micromecanizado.

Respescto al desempeño en el proceso de micromecanizado:

- Las respuestas de los disparos mediante optoacopladores son en general muy buenas, únicamente en la conmutación de 5V a 0 en el caso del láser Spirit se hace un poco lenta, del orden de 6ms.
- La modulación de la potencia en ambos equipos se lleva a cabo de forma correcta.

  Concretamente en el caso del láser Spirit la señal generada por la DAQ Arduino tiene el tiempo de respuesta esperado (del orden de 5ms). La respuesta de la DAQ con respecto a la comunicación con el brazo es bastante buena. El ruido en la salida analógica es aceptable con una amplitud menor que 255mV en el caso más desfavorable. El mayor defecto observado es un pequeño offset, es decir un desplazamiento del cero. Sin embargo, se podría implementar una curva de calibración mediante software para corregir este defecto así como posibles no linealidades.
- En las pruebas de micromecanizado sobre los recortes de papel y aceteto se demuestra que se puede llevar a cabo el proceso de forma correcta.

  Concretando un poco más se observa que la coordinación y respuesta de los sistemas implementados es acorde con la resolución del sistema de translación.

9 ORDEN DE PRIORIDAD ENTRE LOS DOCUMENTOS

Frente a posibles discrepancias se ha de seguir el siguiente orden de prioridad de los documentos:

1. Planos
2. Pliego de Condiciones
3. Presupuesto
4. Memoria
5. Anexos

TÍTULO: **INSTALACIÓN DE UN BRAZO ROBOTIZADO PARA LA AUTOMATIZACIÓN DE LIMPIEZA CON LÁSER DE SUPERFICIES 3D**

ANEXOS

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: JUNIO DE 2019

AUTOR: EL ALUMNO

Fdo.: ADRIÁN CORA SIERRA

Índice del documento ANEXOS

10 DOCUMENTACIÓN DE PARTIDA

10.1. Propuesta inicial de asignación del TFG

ESCUELA UNIVERSITARIA POLITÉCNICA

ASIGNACIÓN DE TRABAJO FIN DE GRADO

En virtud de la solicitud efectuada por:
En virtud da solicitude efectuada por:

APELLIDOS, NOMBRE: *Cora Sierra,Adrián*
APELIDOS E NOME:

DNI: [redacted] **Fecha de Solicitud:** OCT2018
DNI: *Fecha de Solicitude:*

Alumno de esta escuela en la titulación de Grado en Ingeniería en Electrónica Industrial y Automática, se le comunica que la Comisión de Proyectos ha decidido asignarle el siguiente Trabajo Fin de Grado:

O alumno de esta escola na titulación de Grado en Enxeñería en Electrónica Industrial e Automática, comunícaselle que a Comisión de Proxectos ha decidido asignarlle o seguinte Traballo Fin de Grado:

Título T.F.G:Instalación de un brazo robotizado para la automatización de limpieza con láser de superficies en 3D

Número TFG: *770G01A162*

TUTOR:(Titor) *Calvo Rolle,Jose Luis*

COTUTOR/CODIRECTOR: *Alberto Ramil Rego*

La descripción y objetivos del Trabajo son los que figuran en el reverso de este documento*:*

A descrición e obxectivos do proxecto son os que figuran no reverso deste documento.

Ferrol a Jueves, 1 de Noviembre del 2018

Retirei o meu Traballo Fin de Grado o dia _____ de _____ do ano _____

Fdo: Cora Sierra,Adrián

DESCRIPCIÓN Y OBJETIVO:OBJETO: Instalación de un robot de 6 ejes para la manipulación (posicionamiento y orientación) de piezas de pequeño tamaño para ser tratadas por láseres pulsados de alta frecuencia.

ALCANCE: En el laboratorio de Aplicaciones Industriales del Láser del CIT se dispone de dos láseres pulsados de alta frecuencia con los que se realizan diferentes procesos de micromecanizado por ablación (limpieza, texturizado, etc.) Estos equipos disponen de sistemas de posicionamiento que permiten el desplazamiento del haz láser sobre la pieza a tratar. Para aplicar estos procesos a una pieza no plana se hace necesario el poder orientar la pieza de forma que incida perpendicularmente a su superficie. Para ello está previsto la instalación de un pequeño robot industrial que permita la manipulación de las piezas asegurando una velocidad de barrido constante y el mantenimiento de la perpendicularidad entre el haz láser (fijo) y la superficie de la pieza (sujeta por el robot). Este movimiento deberá coordinarse con el inicio/parada de disparo del láser y con el nivel de potencia para lo que se utilizarán las señales de entrada/salida digital/analógica de los diferentes equipos.

Descripción de los hitos previstos del proyecto:
1. Estudio de la documentación técnica de los diversos equipos
2. Diseño de la distribución de los equipos en el laboratorio
3. Análisis de las diferentes posibilidades para el control externo de los láseres
4. Diseño del cableado y de la secuencia de instrucciones que permiten el control externo de los láseres
5. Montaje en el laboratorio
6. Pruebas de desplazamientos y cambios de orientación
7. Evaluación de los resultados de las pruebas
8. Generación de la documentación

11 CÓDIGOS DE PROGRAMACIÓN

11.1. Códigos en Python

Código 11.1: Módulo Python (wizcom.py)

```
"""
Comunicación con un puerto serie

Permite elegir entre:
    - conexión directa por cable con el ordenador
    - a través del micro Wiznet Ethernet-Serial
"""

import socket
import serial  #librería pyserial 3.4
import time

########################################

class Conexion:
    """Conexión a un puerto serie"""
    def __init__(self, SW_ST=False,port=5,baudrate=115200,timeout=0.1,
                 bytesize=8,parity='N',stopbits=1,xonxoff=False,rtscts
                    =False,
                 dsrdtr=False,tcp_ip="192.168.1.127", tcp_port=5000,
                 terminator='<LF>'):
        self.SW_ST = SW_ST
        self.port = port
        self.baudrate = baudrate
        self.timeout = timeout
        self.bytesize = bytesize
        self.parity = parity
        self.stopbits = stopbits
        self.xonxoff = xonxoff
        self.rtscts = rtscts
        self.dsrdtr = dsrdtr
        self.tcp_ip = tcp_ip
        self.tcp_port = tcp_port
        self.terminator = terminator


    def __str__(self):
        msg=('\n  ATRIBUTOS:\n')
        msg=msg+('SW_ST: '+str(self.SW_ST)+'\n')
        msg=msg+('puerto COM: '+str(self.port)+'\n')
        msg=msg+('baudrate: '+str(self.baudrate)+'\n')
        msg=msg+('timeout: '+str(self.timeout)+'\n')
        msg=msg+('bytesize: '+str(self.bytesize)+'\n')
        msg=msg+('parity: '+str(self.parity)+'\n')
```

```
44          msg=msg+('stopbits: '+str(self.stopbits)+'\n')
45          msg=msg+('xonxoff: '+str(self.xonxoff)+'\n')
46          msg=msg+('rtscts: '+str(self.rtscts)+'\n')
47          msg=msg+('dsrdtr: '+str(self.dsrdtr)+'\n')
48          msg=msg+('serv_ip: '+str(self.tcp_ip)+'\n')
49          msg=msg+('port TCP: '+str(self.tcp_port)+'\n')
50          msg=msg+('terminator: '+str(self.terminator)+'\n')
51          return msg
52
53      def open(self):
54          if self.SW_ST==True:
55              #
56              # conexión Serial
57              #
58              ser=serial.Serial(('COM'+str(self.port))) #Sólo vale en
                    windows
59              self.ser = ser
60              ser.baudrate=self.baudrate
61              ser.timeout=self.timeout
62              ser.bytesize=self.bytesize
63              ser.parity=self.parity
64              ser.stopbits=self.stopbits
65              ser.xonxoff=self.xonxoff
66              ser.rtscts=self.rtscts
67              ser.dsrdtr=self.dsrdtr
68
69              time.sleep(0.1)
70              self.estado_SERIAL=True
71
72          else:
73              #
74              # conexión Cliente TCP
75              #
76              sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
77              self.sock = sock
78              sock.settimeout(self.timeout)
79              server_address = (self.tcp_ip, self.tcp_port)
80              sock.connect(server_address)
81
82              time.sleep(0.1)
83              self.estado_TCP=True
84
85      def close(self):
86          if self.SW_ST==True:
87              #
88              # conexión Serial
89              #
90              self.ser.close()
91              self.estado_SERIAL=False
92          else:
93              #
94              # conexión Cliente TCP
95              #
96              self.sock.close()
97              self.estado_TCP=False
98
99      def write(self, message):
100         if self.SW_ST==True:
```

```
101             #
102             # conexión Serial
103             #
104             if (self.estado_SERIAL==True):
105                 if message!=None:
106                     self.ser.write(message.encode("ascii", "ignore"))
107         else:
108             #
109             # conexión Cliente TCP
110             #
111             if(self.estado_TCP==True):
112                 if message!=None:
113                     self.sock.sendall(message.encode("ascii", "ignore"
                        ))
114
115
116     def read(self):
117         linea=self.readline()
118         return linea.rstrip()
119
120     def readline(self):
121         # readline se define para mantener compatibilidad con los
                programas
122         # anteriores que usaban la librería PySerial
123         if self.SW_ST==True:
124             #
125             # conexión Serial
126             #
127             if (self.estado_SERIAL==True):
128                 try:
129                     linea=self.ser.readline()
130                     linea=str(linea)
131                     linea=linea[2:-1]
132                     linea=linea.replace('\\n','\n')
133                     linea=linea.replace('\\r','\r')
134                     return str(linea)
135                 except:
136                     return '' #para que no se pare en el error de
                            timeout
137             else:
138                 linea=''
139                 return linea
140         else:
141             #
142             # conexión Cliente TCP
143             #
144              if(self.estado_TCP==True):
145                 try:
146                     response = self.sock.recv(1024)
147                     response=str(response)
148                     response=response[2:-1]
149                     response=response.replace('\\n','\n')
150                     response=response.replace('\\r','\r')
151                     return str(response)
152                 except:
153                     return '' #para que no se pare en el error de
                            timeout
154              else:
```

```
155                 linea=''
156                 return linea
157
158     def aplicaConfiguracion(self, fileName='Configuracion.txt'):
159         #
160         #   LEE FICHERO CONFIGURACION Y ACTUALIZA VALORES
161         try:
162             with open(fileName, 'r') as f:
163                 for line in f:
164                     line = line.rstrip()
165                     if line!='' :
166                         print(line)
167                         variable = line.split('=')[0].lstrip().rstrip
                            ()
168                         valor = line.split('=')[1].lstrip().rstrip()
169                         self.__dict__[variable] = valor
170             #
171             #   ASIGNA LOS TIPOS int, float, logical
172             if self.SW_ST=='True':
173                 self.SW_ST=True
174             else:
175                 self.SW_ST=False
176             self.port=int(self.port)
177             self.baudrate=int(self.baudrate)
178             self.timeout=float(self.timeout)
179             self.bytesize=int(self.bytesize)
180             self.stopbits=int(self.stopbits)
181             if self.xonxoff=='False':
182                 self.xonxoff=False
183             else:
184                 self.xonxoff=True
185             if self.rtscts=='False':
186                 self.rtscts=False
187             else:
188                 self.rtscts=True
189             if self.dsrdtr=='False':
190                 self.dsrdtr=False
191             else:
192                 self.dsrdtr=True
193             self.tcp_port=int(self.tcp_port)
194         except:
195             pass
196
197     def salvaConfiguracion(self, fileName='Configuracion.txt'):
198         diccionario = self.__dict__
199         f = open(fileName, mode='w')
200         nb = 0
201         for variable in diccionario.keys():
202             cadena = variable + " = " + str(diccionario[variable]) + '
                    \r\n'
203             nb = nb + f.write(cadena)
204         f.close()
205         print('{} bytes escritos en el fichero {}'.format(nb, fileName
                ))
206
207 ########################################
208 #
209 #   MAIN
```

```
210 #
211 ########################################
212
213 if __name__ == "__main__":
214     import os
215     name = 'ConfiguracionNew.txt'
216     workDir = 'C:\\Users\\usuario\\Documents\\laserAVIA\\python3'
217     workDir = '.'
218     workDir = os.path.abspath(os.path.curdir)
219     fileName = os.path.join(workDir, name)
220     #
221     #   CREA CONEXION CON LOS VALORES POR DEFECTO
222     Datos = Conexion()
223     print( vars(Datos) )
224     #
225     #   LEE FICHERO CONFIGURACION Y ACTUALIZA VALORES
226     Datos.aplicaConfiguracion()
227     #
228     #   MODIFICO VALORES
229 #     print(Datos.SW_ST)
230 #     print("==>")
231 #     if Datos.SW_ST == True:
232 #         print("hola que tal")
233     Datos.SW_ST = True
234     Datos.port = 1
235     Datos.baudrate = 9600
236     Datos.timeout = 0.1
237     Datos.tcp_ip = '193.144.52.53'
238     Datos.tcp_port = 80
239     #
240     #   ESCIRBO FICHERO CONFIGURACION
241     Datos.salvaConfiguracion(fileName)
242
243     print(str(Datos))
```

Código 11.2: Interface ventana de comandos y monitor en Python (Interface.py)

```
#INTERFACE GRAFICA MONITOR
from tkinter import *
from tkinter import messagebox #messagebox.showinfo("Title","abcd")
from tkinter import filedialog
import os

import wizcom

class App_Monitor(Frame):
    def __init__(self,master,COM):
        super(App_Monitor,self).__init__(master)
        self.grid()
        self.COM=COM
        self.crear_widgets()
        master.title('Cliente TCP/Serial')
        self.monitor.after(1000,self.leer_TCP)

    def crear_widgets(self):
        #monitor
        self.monitor=Text(self, width=95, height=15)
        self.monitor.grid(row=1,columnspan=4)
        self.monitor_scroll=Scrollbar(self,width=16,takefocus=1)
        self.monitor_scroll.grid(row=0,rowspan=4,column=4,sticky=N+S)
        self.monitor_scroll.config( command = self.monitor.yview )
        self.monitor.configure(yscrollcommand=self.monitor_scroll.set)
        #botones
        self.boton_enviar=Button(self,text='Enviar',command=self.
            enviar_TCP)
        self.boton_enviar.grid(row=2,column=3,sticky=W+E)

        self.boton_limpiar=Button(self,text='Limpiar Pantalla',command
            =self.desborde)
        self.boton_limpiar.grid(row=0,column=1,sticky=E)

        self.boton_reset=Button(self,text='Reset App Monitor',command=
            self.reset_APP)
        self.boton_reset.grid(row=0,column=2,sticky=W)

        self.boton_reset=Button(self,text='Abrir Script',command=self.
            abrir_script)
        self.boton_reset.grid(row=0,column=3,sticky=E)
        #entrada
        self.texto_enviar=Entry(self,text='',width=110)
        self.texto_enviar.grid(row=2,column=0,columnspan=3)
        self.texto_enviar.bind('<Return>', self.return_enviar_TCP)


    def desborde(self):
        self.monitor.delete('1.0', 'end')

    def enviar_TCP(self):
        mensaje=self.texto_enviar.get()
        if self.COM.terminator=='<CR>' :
            terminator='\r'
        else:
```

```
52              if self.COM.terminator=='<LF><CR>' :
53                  terminator='\n\r'
54              else:
55                  if self.COM.terminator=='<CR><LF>' :
56                      terminator='\n\r'
57                  else:
58                      terminator='\n'
59          self.monitor.insert('insert', "-->: "+mensaje+"\n")
60          self.monitor.see("end")
61          mensaje=mensaje+terminator
62          self.COM.write(message=mensaje)
63          self.texto_enviar.text=('')
64
65      def return_enviar_TCP(self,event):
66          self.enviar_TCP()
67
68      def leer_TCP(self):
69          mensaje=self.COM.read()
70          if mensaje!='':
71              self.monitor.insert('insert', "<--: "+mensaje+"\n")
72              self.monitor.see("end")
73          self.monitor.after(1000,self.leer_TCP)
74
75      def reset_APP(self):
76          global cerrar_APP
77          cerrar_APP=False
78          root.destroy()
79
80      def abrir_script(self):
81          global script,cerrar_APP
82          cerrar_APP=True
83          script=True
84          root.destroy()
85
86  class App_PedirDatos(Frame):
87      def __init__(self,master,COM):
88          super(App_PedirDatos,self).__init__(master)
89          self.grid()
90          self.COM=COM
91          self.crear_widgets()
92          master.title('Datos: Cliente TCP/Serial')
93
94      def crear_widgets(self):
95          #global Datos
96          #RADIAL SWITCH
97          self.l1=Label(self,text='TIPO DE CONEXIÓN:').grid(sticky=W,
                columnspan=3)
98          self.SW_ST=StringVar() #False==>TCP
99          self.SW_ST.set('TCP')
100         self.r1=Radiobutton(self,text='Cliente TCP',
101                             variable=self.SW_ST,value='TCP')
102         self.r1.grid(sticky=W,column=1)
103         self.r2=Radiobutton(self,text='Serial',
104                             variable=self.SW_ST,value='SERIAL')
105         self.r2.grid(sticky=W,column=1)
106
107
108         self.l2=Label(self,text='PARÁMETROS TCP:').grid(sticky=W,
```

```
            columnspan=3)
109
110         self.l21=Label(self,text='Timeout:').grid(sticky=W,column=1)
111
112         self.Ttimeout=Entry(self)
113         self.Ttimeout.grid(sticky=W,column=2,row=4)
114         self.Ttimeout.delete(0, END)
115         self.Ttimeout.insert(0, str(self.COM.timeout))
116
117         self.l22=Label(self,text='Server IP:').grid(sticky=W,column=1)
118
119         self.tcp_ip=Entry(self)
120         self.tcp_ip.grid(sticky=W,column=2,row=5)
121         self.tcp_ip.delete(0, END)
122         self.tcp_ip.insert(0, str(self.COM.tcp_ip))
123
124         self.l23=Label(self,text='Port:').grid(sticky=W,column=1)
125
126         self.tcp_port=Entry(self)
127         self.tcp_port.grid(sticky=W,column=2,row=6)
128         self.tcp_port.delete(0, END)
129         self.tcp_port.insert(0, str(self.COM.tcp_port))
130
131         self.l3=Label(self,text='PARÁMETROS SERIAL:').grid(sticky=W,
            columnspan=2)
132
133         self.l31=Label(self,text='Timeout:').grid(sticky=W,column=1)
134
135         self.Stimeout=Entry(self)
136         self.Stimeout.grid(sticky=W,column=2,row=8)
137         self.Stimeout.delete(0, END)
138         self.Stimeout.insert(0, str(self.COM.timeout))
139
140         self.l32=Label(self,text='Puerto:').grid(sticky=W,column=1)
141
142         self.port=Entry(self)
143         self.port.grid(sticky=W,column=2,row=9)
144         self.port.delete(0, END)
145         self.port.insert(0, str(self.COM.port))
146
147         self.l33=Label(self,text='Baudrate:').grid(sticky=W,column=1)
148
149         self.baudrate=Entry(self)
150         self.baudrate.grid(sticky=W,column=2,row=10)
151         self.baudrate.delete(0, END)
152         self.baudrate.insert(0, str(self.COM.baudrate))
153
154         self.l34=Label(self,text='Tamaño de byte:').grid(sticky=W,
            column=1)
155
156         self.bytesize=Entry(self)
157         self.bytesize.grid(sticky=W,column=2,row=11)
158         self.bytesize.delete(0, END)
159         self.bytesize.insert(0, str(self.COM.bytesize))
160
161         self.l35=Label(self,text='Paridad:').grid(sticky=W,column=1)
162
163         self.parity=Entry(self)
```

```
164         self.parity.grid(sticky=W,column=2,row=12)
165         self.parity.delete(0, END)
166         self.parity.insert(0, str(self.COM.parity))
167 
168         self.l36=Label(self,text='Bits de Parada:').grid(sticky=W,
                column=1)
169 
170         self.stopbits=Entry(self)
171         self.stopbits.grid(sticky=W,column=2,row=13)
172         self.stopbits.delete(0, END)
173         self.stopbits.insert(0, str(self.COM.stopbits))
174 
175         self.l37=Label(self,text='xonxoff:').grid(sticky=W,column=1)
176 
177         self.xonxoff=Entry(self)
178         self.xonxoff.grid(sticky=W,column=2,row=14)
179         self.xonxoff.delete(0, END)
180         self.xonxoff.insert(0, str(self.COM.xonxoff))
181 
182         self.l38=Label(self,text='rtscts:').grid(sticky=W,column=1)
183 
184         self.rtscts=Entry(self)
185         self.rtscts.grid(sticky=W,column=2,row=15)
186         self.rtscts.delete(0, END)
187         self.rtscts.insert(0, str(self.COM.rtscts))
188 
189         self.l39=Label(self,text='dsrdtr:').grid(sticky=W,column=1)
190 
191         self.dsrdtr=Entry(self)
192         self.dsrdtr.grid(sticky=W,column=2,row=16)
193         self.dsrdtr.delete(0, END)
194         self.dsrdtr.insert(0, str(self.COM.dsrdtr))
195 
196         self.l3=Label(self,text='PARÁMETROS COMUNES:').grid(sticky=W,
                columnspan=2)
197 
198         self.l39=Label(self,text='Terminador\n[<CR>,<LF>,<CR><LF>,<LF
                ><CR>]:').grid(sticky=W,column=1)
199 
200         self.terminador=Entry(self)
201         self.terminador.grid(sticky=W,column=2,row=18)
202         self.terminador.delete(0, END)
203         self.terminador.insert(0, str(self.COM.terminator))
204 
205         #Botones:
206         self.boton_1=Button(self,text='Aceptar',command=lambda : self.
                exit_APP())
207         self.boton_1.grid(row=19,column=2,sticky=E)
208         self.boton_2=Button(self,text='Salvar Datos',command=lambda :
                self.s_datos())
209         self.boton_2.grid(row=19,column=1,sticky=E)
210 
211         self.after(1000,self.switch_a)
212 
213     def s_datos(self):
214         self.adquirir_Datos()
215         self.COM.salvaConfiguracion()
216         messagebox.showinfo("Datos salvados",str(self.COM))
```

```
217
218     def exit_APP(self):
219         global cerrar_APP
220         cerrar_APP=False
221         self.adquirir_Datos()
222         self.COM.salvaConfiguracion()
223         root.destroy()
224
225     def adquirir_Datos(self):
226         #global Datos
227         Datos=self.COM
228         Datos.port=self.port.get()
229         Datos.baudrate=self.baudrate.get()
230         if Datos.SW_ST==False:#False==>TCP
231             Datos.timeout=self.Ttimeout.get()
232         else:
233             Datos.timeout=self.Stimeout.get()
234         Datos.bytesize=self.bytesize.get()
235         Datos.parity=self.parity.get()
236         Datos.stopbits=self.stopbits.get()
237         Datos.xonxoff=self.xonxoff.get()
238         Datos.rtscts=self.rtscts.get()
239         Datos.dsrdtr=self.dsrdtr.get()
240         Datos.tcp_ip=self.tcp_ip.get()
241         Datos.tcp_port=self.tcp_port.get()
242         Datos.terminator=self.terminador.get()
243
244         Datos.port=int(Datos.port)
245         Datos.baudrate=int(Datos.baudrate)
246         Datos.timeout=float(Datos.timeout)
247         Datos.bytesize=int(Datos.bytesize)
248         Datos.stopbits=int(Datos.stopbits)
249
250         if Datos.xonxoff=='False':
251             Datos.xonxoff=False
252         else:
253             Datos.xonxoff=True
254
255         if Datos.rtscts=='False':
256             Datos.rtscts=False
257         else:
258             Datos.rtscts=True
259
260         if Datos.dsrdtr=='False':
261             Datos.dsrdtr=False
262         else:
263             Datos.dsrdtr=True
264
265         Datos.tcp_port=int(Datos.tcp_port)
266
267     def switch_a(self): #False==>TCP
268         Datos=self.COM
269         if str(self.SW_ST.get())=='SERIAL':
270             Datos.SW_ST=True
271         else:
272             Datos.SW_ST=False
273         self.COM=Datos
274         self.after(500,self.switch_a)
```

```
275
276 #ABRIR UN SCRIPT nombre.py CON DIALOG BOX
277 def script_dialog(root):
278     original_path=os.getcwd()
279     root.withdraw()
280     file_path = filedialog.askopenfilename()
281     nombre=file_path
282
283     i=len(nombre)
284     a=''
285     while(a!='/'):
286         i-=1
287         a=nombre[i]
288
289     nombre=nombre[(i+1):-3]
290
291     file_path=file_path[:i]
292     os.chdir(file_path.replace("/", "\\"))
293
294     try:
295         print('Ejecutar script')
296         __import__(nombre)
297         print('Script Ejecutado')
298     except:
299         pass
300
301     os.chdir(original_path)
302
303 #INICIALIZACIÓN Y CIERRE DE LA APP
304 if ( __name__=='__main__' ):
305     global Cerrar_APP
306     cerrar_APP=False
307
308     while (cerrar_APP==False):
309         cerrar_APP=True
310
311         root=Tk()
312         root.geometry("350x450")
313         COM=wizcom.Conexion()
314         COM.aplicaConfiguracion()
315         app=App_PedirDatos(root,COM)
316         root.mainloop()
317
318
319         if cerrar_APP==False:
320             cerrar_APP=True
321             COM=wizcom.Conexion()
322             COM.aplicaConfiguracion()
323             COM.open(); print('Puertos_abiertos')
324             global script
325             script=False
326             root=Tk()
327             app=App_Monitor(root,COM)
328             root.mainloop()
329             COM.close();
330             if (script==True):
331                 root1 = Tk()
332                 script_dialog(root=root1)
```

```
333                     root1.destroy()
334                 print('Puertos_cerrados')
```

11.2. Códigos de Arduino IDE

Código 11.3: DAQ Arduino (Main.ino)

```
  breakatwhitespace
//Variables lectura comandos:
  char comando[15];
  int n_comando;
//Comandos:
  const char AO_PWM_0[6]="SPWM0";
  const char AO_PWM_1[6]="SPWM1";
  const char AO_PWM_2[6]="SPWM2";
  const char AO_PWM_3[6]="SPWM3";
  const char AO_PWM_4[6]="SPWM4";
  const char AO_PWM_5[6]="SPWM5";

  const char AI_Read_0[6]="SRAI0";
  const char AI_Read_1[6]="SRAI1";
  const char AI_Read_2[6]="SRAI2";
  const char AI_Read_3[6]="SRAI3";
  const char AI_Read_4[6]="SRAI4";
  const char AI_Read_5[6]="SRAI5";

void setup() {
  //Pines PWM en modo salida
    pinMode(11, OUTPUT);
    pinMode(10, OUTPUT);
    pinMode(9, OUTPUT);
    pinMode(6, OUTPUT);
    pinMode(5, OUTPUT);
    pinMode(3, OUTPUT);

  //Timers en modo PWM Fast, PWM 8bits (T_PWM=min)
    //TIMER0
      TCCR0A=TCCR0A&0b10101111;
      TCCR0A=TCCR0A|0b10100011;
      TCCR0B=TCCR0B&0b00110001;
      TCCR0B=TCCR0B|0b00000001;
    //TIMER1
      TCCR1A=TCCR1A&0b10101101;
      TCCR1A=TCCR1A|0b10100001;
      TCCR1B=TCCR1B&0b00101001;
      TCCR1B=TCCR1B|0b00001001;
      TCCR1C=TCCR1C&0b00111111;
      TCCR1C=TCCR1C|0b00000000;
    //TIMER2
      TCCR2A=TCCR2A&0b10101111;
```

```
43        TCCR2A=TCCR2A|0b10100011;
44        TCCR2B=TCCR2B&0b00110001;
45        TCCR2B=TCCR2B|0b00000001;
46
47    //Abrir puerto
48      Serial.begin(9600);
49  }
50
51  void loop() {
52    if(Serial.available()){
53      //Lectura de comando:
54        n_comando=Serial.readBytesUntil('\r', comando, (sizeof(comando) - 1));
55        comando[n_comando]='\0'; //caracter nulo al final necesario en C
56        //Serial.read(); // Descarta el '\n' que suele venir con el '\r'
57
58      //Evaluación del comando:
59        int task=0,valor=0;
60        task=check_task(comando,&valor);
61        /*Serial.print("Tarea: ");
62        Serial.println(task);
63        Serial.print("Valor: ");
64        Serial.println(valor);*/
65        if ((task>=10)&&(task<20))
66        {
67          set_PIN_PWM(task,valor); //función de generación de salidas PWM
68          Serial.print("SPWM");
69          Serial.print((task-10));
70          Serial.print("\n\r");
71          Serial.print(valor);
72          Serial.print("\n\r");
73        }else if ((task>=20)&&(task<30))
74        {
75          valor=read_AI(task); //función de lectura entradas analógicas
76          Serial.print("SRAI");
77          Serial.print((task-20));
78          Serial.print("\n\r");
79          Serial.print(valor);
80          Serial.print("\n\r");
81        }
82    }
83  }
84
85  int read_AI(int task)
86  {
87    int valor_int=0;
88    if (task==20) //PINA0
89    {
90      valor_int=analogRead(A0);
91    }
92    else if (task==21) //PINA1
```

```
93    {
94      valor_int=analogRead(A1);
95    }
96    else if (task==22) //PINA2
97    {
98      valor_int=analogRead(A2);
99    }
100   else if (task==23) //PINA3
101   {
102     valor_int=analogRead(A3);
103   }
104   else if (task==24) //PINA4
105   {
106     valor_int=analogRead(A4);
107   }
108   else if (task==25) //PINA5
109   {
110     valor_int=analogRead(A5);
111   }
112   return valor_int;
113 }
114
115 void set_PIN_PWM(int task ,int valor)
116 {
117   if ((valor>=0)&&(valor<=255))
118   {
119     if (task==10) //PIN11
120     {
121       OCR2A=valor;
122     }
123     else if (task==11) //PIN10
124     {
125       OCR1B=valor;
126     }
127     else if (task==12) //PIN9
128     {
129       OCR1A=valor;
130     }
131     else if (task==13) //PIN6
132     {
133       OCR0A=valor;
134     }
135     else if (task==14) //PIN5
136     {
137       OCR0B=valor;
138     }
139     else if (task==15) //PIN3
140     {
141       OCR2B=valor;
142     }
```

```
143   }
144 }
145
146 int check_task(char comando[],int *valor)
147 {
148   char valor_char[5];
149   int valor_int,task=0;
150   /* task==0 ninguna tarea
151    *
152    * task==10 generación salida analógica PWM 0
153    * task==11 generación salida analógica PWM 1
154    * ...
155    * task==20 lectura entrada analógica 0
156    * task==21 lectura entrada analógica 1
157    * ...
158    */
159
160   if (check_comand(comando,AO_PWM_0)==true)
161     {
162       int n_char=0;
163
164       while(Serial.available()<=0);
165       n_char=Serial.readBytesUntil('\r', valor_char, (sizeof(valor_char) - 1));
166       valor_char[n_char]='\0';
167       valor_int=atol(valor_char); //atol() función de conversión de char a int
168       if ((valor_int>=0)&&(valor_int<=255))
169       {
170         *valor=valor_int;
171         task=10;
172       }
173     }
174   else if (check_comand(comando,AO_PWM_1)==true)
175     {
176       int n_char=0;
177
178       while(Serial.available()<=0);
179       n_char=Serial.readBytesUntil('\r', valor_char, (sizeof(valor_char) - 1));
180       valor_char[n_char]='\0';
181       valor_int=atol(valor_char); //atol() función de conversión de char a int
182       if ((valor_int>=0)&&(valor_int<=255))
183       {
184         *valor=valor_int;
185         task=11;
186       }
187     }
188   else if (check_comand(comando,AO_PWM_2)==true)
189     {
190       int n_char=0;
191
192       while(Serial.available()<=0);
```

```
193       n_char=Serial.readBytesUntil('\r', valor_char, (sizeof(valor_char) - 1));
194       valor_char[n_char]='\0';
195       valor_int=atol(valor_char); //atol() función de conversión de char a int
196       if ((valor_int>=0)&&(valor_int<=255))
197       {
198         *valor=valor_int;
199         task=12;
200       }
201     }
202   else if (check_comand(comando,AO_PWM_3)==true)
203     {
204       int n_char=0;
205
206       while(Serial.available()<=0);
207       n_char=Serial.readBytesUntil('\r', valor_char, (sizeof(valor_char) - 1));
208       valor_char[n_char]='\0';
209       valor_int=atol(valor_char); //atol() función de conversión de char a int
210       if ((valor_int>=0)&&(valor_int<=255))
211       {
212         *valor=valor_int;
213         task=13;
214       }
215     }
216   else if (check_comand(comando,AO_PWM_4)==true)
217     {
218       int n_char=0;
219
220       while(Serial.available()<=0);
221       n_char=Serial.readBytesUntil('\r', valor_char, (sizeof(valor_char) - 1));
222       valor_char[n_char]='\0';
223       valor_int=atol(valor_char); //atol() función de conversión de char a int
224       if ((valor_int>=0)&&(valor_int<=255))
225       {
226         *valor=valor_int;
227         task=14;
228       }
229     }
230   else if (check_comand(comando,AO_PWM_5)==true)
231     {
232       int n_char=0;
233
234       while(Serial.available()<=0);
235       n_char=Serial.readBytesUntil('\r', valor_char, (sizeof(valor_char) - 1));
236       valor_char[n_char]='\0';
237       valor_int=atol(valor_char); //atol() función de conversión de char a int
238       if ((valor_int>=0)&&(valor_int<=255))
239       {
240         *valor=valor_int;
241         task=15;
242       }
```

```
243       }
244     else if (check_comand(comando,AI_Read_0)==true)
245     {
246       task=20;
247     }
248     else if (check_comand(comando,AI_Read_1)==true)
249     {
250       task=21;
251     }
252     else if (check_comand(comando,AI_Read_2)==true)
253     {
254       task=22;
255     }
256     else if (check_comand(comando,AI_Read_3)==true)
257     {
258       task=23;
259     }
260     else if (check_comand(comando,AI_Read_4)==true)
261     {
262       task=24;
263     }
264     else if (check_comand(comando,AI_Read_5)==true)
265     {
266       task=25;
267     }
268     return task;
269   }
270   bool check_comand(char in_comand[],const char const_comand[])
271   {
272     bool check=true;
273     int i=0;
274
275     while(in_comand[i+1]!='\0'){
276       if (in_comand[i]!=const_comand[i])
277         {
278           //Serial.print(i);
279           //Serial.println(" false");
280           check=false;
281         }
282       i++;
283     }
284     if(const_comand[i]!='\0'){
285       //Serial.println("n distinto");
286       check=false;
287     }
288     return check;
289   }
```

11.3. Códigos en RAPID

Código 11.4: Módulo RAPID modulación de potencia AVIA (AviaTcp.mod)

```
%%%
    VERSION: 1
    LANGUAGE: ENGLISH
%%%

MODULE AviaTcp
    !
    !   Module for communication with the AVIA laser via TCP / IP
    !

    RECORD tcp_connection
        socketdev socket;
        string ip;
        num port;
        string terminator;
        string status;
        ! Data record for TCP/IP connection
    ENDRECORD

    VAR tcp_connection AVIA;

    PROC avia_close()
        ! Close the socket
        SocketClose AVIA.socket;
        AVIA.status:="Close";
    ENDPROC

    PROC avia_open()
        ! Create and connect the socket
        AVIA.ip:="10.113.16.50";
        AVIA.port:=5000;
        AVIA.terminator:="\0D\0A";
        SocketCreate AVIA.socket;
        SocketConnect AVIA.socket,AVIA.ip,AVIA.port;
        AVIA.status:="Open";
    ERROR
        TPWrite "Error opening the socket";
        avia_close;
        EXIT;
    ENDPROC

    FUNC string avia_command(string token)
        VAR string receive_string;
        !   Send message
        SocketSend AVIA.socket\Str:=(token+AVIA.terminator);
        WaitTime 0.1;
        !   Receive message
        SocketReceive AVIA.socket\Str:=receive_string;
        RETURN receive_string;
    ERROR
```

```
51          RETURN "";
52      ENDFUNC
53
54      PROC avia_test()
55          VAR string receive_string;
56          receive_string:=avia_command(">=0");
57          receive_string:=avia_command("E=0A");
58          !         TPWrite "=>?HSN";
59          receive_string:=avia_command("?HSN");
60          !         TPWrite "<="+receive_string;
61          IF receive_string=("7696.00"+AVIA.terminator) THEN
62              TPWrite "AVIA found in the port "+NumToStr(AVIA.port,0);
63              RETURN ;
64          ELSE
65              TPWrite "<="+receive_string;
66              TPWrite "AVIA does not respond at the port "+NumToStr(AVIA
                    .port,0);
67              avia_close;
68              EXIT;
69          ENDIF
70      ENDPROC
71
72      FUNC bool avia_shutter_open()
73          VAR string receive_string;
74          VAR bool ok;
75          VAR num nval;
76          !   Opens shutter
77          receive_string:=avia_command("S=1");
78          WaitTime 0.2;
79          receive_string:=avia_command("?S");
80          ok:=StrToVal(receive_string,nval);
81          IF ok THEN
82              IF nval=1 THEN
83                  RETURN TRUE;
84              ELSE
85                  TPWrite "ERROR: can't open Shutter";
86                  RETURN FALSE;
87              ENDIF
88          ELSE
89              TPWrite "<="+receive_string;
90              TPWrite "WARNING: bad Shutter Open response";
91              RETURN FALSE;
92          ENDIF
93      ENDFUNC
94
95      FUNC bool avia_shutter_close()
96          VAR string receive_string;
97          VAR bool ok;
98          VAR num nval;
99          !   Closes shutter
100         receive_string:=avia_command("S=0");
101         WaitTime 0.2;
102         receive_string:=avia_command("?S");
103         ok:=StrToVal(receive_string,nval);
104         IF ok THEN
105             IF nval=0 THEN
106                 RETURN TRUE;
107             ELSE
```

```
108                 TPWrite "ERROR: can't close Shutter";
109                 RETURN FALSE;
110             ENDIF
111         ELSE
112             TPWrite "<="+receive_string;
113             TPWrite "WARNING: bad Shutter Close response";
114             RETURN FALSE;
115         ENDIF
116     ENDFUNC
117 
118     FUNC bool avia_diodes_on()
119         VAR string receive_string;
120         VAR bool ok;
121         VAR num nval;
122         !   Diodes current ON
123         receive_string:=avia_command("DO=1");
124         WaitTime 0.5;
125         receive_string:=avia_command("?DO");
126         ok:=StrToVal(receive_string,nval);
127         IF ok THEN
128             IF nval=1 THEN
129                 RETURN TRUE;
130             ELSE
131                 TPWrite "ERROR: can't ON Diodes";
132                 RETURN FALSE;
133             ENDIF
134         ELSE
135             TPWrite "<="+receive_string;
136             TPWrite "WARNING: bad Diodes ON response";
137             RETURN FALSE;
138         ENDIF
139     ENDFUNC
140 
141     FUNC bool avia_diodes_off()
142         VAR string receive_string;
143         VAR bool ok;
144         VAR num nval;
145         !   Diodes Current OFF
146         receive_string:=avia_command("DO=0");
147         WaitTime 0.5;
148         receive_string:=avia_command("?DO");
149         ok:=StrToVal(receive_string,nval);
150         IF ok THEN
151             IF nval=0 THEN
152                 RETURN TRUE;
153             ELSE
154                 TPWrite "ERROR: can't OFF Diodes";
155                 RETURN FALSE;
156             ENDIF
157         ELSE
158             TPWrite "<="+receive_string;
159             TPWrite "WARNING: bad Diodes OFF response";
160             RETURN FALSE;
161         ENDIF
162     ENDFUNC
163 
164     FUNC bool avia_pulsing_on()
165         VAR string receive_string;
```

```
166         VAR bool ok;
167         VAR num nval;
168         !   Pulsing Control ON
169         receive_string:=avia_command("PC=1");
170         WaitTime 0.5;
171         receive_string:=avia_command("?PC");
172         ok:=StrToVal(receive_string,nval);
173         IF ok THEN
174             IF nval=1 THEN
175                 RETURN TRUE;
176             ELSE
177                 TPWrite "ERROR: can't Pulsing Control ON";
178                 RETURN FALSE;
179             ENDIF
180         ELSE
181             TPWrite "<="+receive_string;
182             TPWrite "WARNING: bad Pulsing Control ON response";
183             RETURN FALSE;
184         ENDIF
185     ENDFUNC
186
187     FUNC bool avia_pulsing_off()
188         VAR string receive_string;
189         VAR bool ok;
190         VAR num nval;
191         !   Pulsing Control OFF
192         receive_string:=avia_command("PC=0");
193         WaitTime 0.5;
194         receive_string:=avia_command("?PC");
195         ok:=StrToVal(receive_string,nval);
196         IF ok THEN
197             IF nval=0 THEN
198                 RETURN TRUE;
199             ELSE
200                 TPWrite "ERROR: can't Pulsing Control OFF";
201                 RETURN FALSE;
202             ENDIF
203         ELSE
204             TPWrite "<="+receive_string;
205             TPWrite "WARNING: bad Pulsing Control OFF response";
206             RETURN FALSE;
207         ENDIF
208     ENDFUNC
209
210     FUNC bool avia_set_frequency(num freq)
211         VAR string send_string;
212         VAR string receive_string;
213         VAR bool ok;
214         VAR num nval;
215         !   Sets Repetition Rate in Hz
216         IF (freq<3) OR (freq>100000) THEN
217             TPWrite "ERROR: The frequency must be between 3Hz and 100
                        kHz";
218             RETURN FALSE;
219         ENDIF
220         send_string:="RR="+NumToStr(freq,0);
221         receive_string:=avia_command(send_string);
222         WaitTime 0.2;
```

```
223         receive_string:=avia_command("?RR");
224         ok:=StrToVal(receive_string,nval);
225         IF ok THEN
226             IF nval=freq THEN
227                 RETURN TRUE;
228             ELSE
229                 TPWrite "ERROR: can't set Repetition Rate";
230                 RETURN FALSE;
231             ENDIF
232         ELSE
233             TPWrite "<="+receive_string;
234             TPWrite "WARNING: bad Repetition Rate response";
235             RETURN FALSE;
236         ENDIF
237     ENDFUNC
238
239     FUNC bool avia_set_current(num current)
240         VAR string send_string;
241         VAR string receive_string;
242         VAR bool ok;
243         VAR num cval;
244         !   Sets Repetition Rate in Hz
245         IF (current<0) OR (current>100) THEN
246             TPWrite "ERROR: The Diode Current percentage must be
                        between 0 and 100";
247             RETURN FALSE;
248         ENDIF
249         send_string:="C="+NumToStr(current,1);
250         receive_string:=avia_command(send_string);
251         WaitTime 0.2;
252         receive_string:=avia_command("?CPT");
253         ok:=StrToVal(receive_string,cval);
254         IF ok THEN
255             IF Abs(cval-current)<0.1 THEN
256                 RETURN TRUE;
257             ELSE
258                 TPWrite "Current= "\Num:=cval;
259                 TPWrite "ERROR: can't set Diode Current";
260                 RETURN FALSE;
261             ENDIF
262         ELSE
263             TPWrite "<="+receive_string;
264             TPWrite "WARNING: bad Diode Current response";
265             RETURN FALSE;
266         ENDIF
267     ENDFUNC
268
269     FUNC bool avia_set_pulse_mode(num mode)
270         VAR string send_string;
271         VAR string receive_string;
272         VAR bool ok;
273         VAR num nval;
274         !   Sets Pulse Mode
275         !   0=CONTINUOUS, 1=BURST, 2=ALIGNMENT
276         IF (mode<0) OR (mode>2) THEN
277             TPWrite "ERROR: Pulse Mode must be 0, 1 or 2";
278             RETURN FALSE;
279         ENDIF
```

```
280         send_string:="PM="+NumToStr(mode,0);
281         receive_string:=avia_command(send_string);
282         WaitTime 0.5;
283         receive_string:=avia_command("?PM");
284         ok:=StrToVal(receive_string,nval);
285         IF ok THEN
286             IF nval=mode THEN
287                 RETURN TRUE;
288             ELSE
289                 TPWrite "ERROR: can't set Pulse Mode";
290                 RETURN FALSE;
291             ENDIF
292         ELSE
293             TPWrite "<="+receive_string;
294             TPWrite "WARNING: bad Pulse Mode response";
295             RETURN FALSE;
296         ENDIF
297     ENDFUNC
298
299
300 ENDMODULE
```

Código 11.5: Módulo RAPID DAQ Arduino (SpiritArduino.mod)

```
%%%
    VERSION: 1
    LANGUAGE: ENGLISH
%%%


MODULE SpiritArduino
    !
    !   Module for e handling of the SPIRIT laser
    !
    RECORD Arduino_handling
        iodev channel;
        string terminator;
        string status;
        ! Data record for Arduino handling
    ENDRECORD

    VAR Arduino_handling SPIRIT;

    VAR bool iprint:=FALSE;
    VAR string query;
    VAR string answer;



    !============================
    ! OPEN SERIAL PORT
    !============================
    PROC spirit_open()
        Open "com1:",SPIRIT.channel\Bin;
        SPIRIT.status:="Open";
        !!!SPIRIT.terminator:="\0D\0A\0D";
        SPIRIT.terminator:="\0A\0D";
        IF iprint THEN
            TPWrite "The COM port is open";
        ENDIF
    ERROR
        TPWrite "ERROR: can't open the port";
        EXIT;
    ENDPROC

    !============================
    ! CLOSE SERIAL PORT
    !============================
    PROC spirit_close()
        Close SPIRIT.channel;
        SPIRIT.status:="Close";
        IF iprint THEN
            TPWrite "The COM port is closed";
        ENDIF
    ERROR
        TPWrite "ERROR: can't close the port";
        EXIT;
    ENDPROC

```

```
56      !============================
57      ! set analog output for laser power
58      !============================
59      FUNC bool spirit_set_power(num power)
60          VAR num canal_PWM:=0;
61          VAR num volts;
62          VAR num PWM_level;
63          VAR num response_level;
64          !
65          !   Sets Output Power in percentage
66          !
67          IF (power<0) OR (power>100) THEN
68              TPWrite "ERROR: The Diode Current percentage must be
                    between 0 and 100";
69              RETURN FALSE;
70          ENDIF
71          !
72          volts:=5.0*power/100;
73          PWM_level:=Round(volts/5.0*255);
74          response_level:=arduinoPWM(canal_PWM,PWM_level);
75          IF response_level=PWM_level THEN
76              RETURN TRUE;
77          ELSE
78              IF response_level<0 THEN
79                  TPWrite "ERROR: leyendo PWM, val = "+NumToStr(
                        response_level,0);
80                  RETURN FALSE;
81              ELSE
82                  TPWrite "ERROR: escribiendo PWM, val = "+NumToStr(
                        response_level,0);
83                  RETURN FALSE;
84              ENDIF
85          ENDIF
86      ENDFUNC
87
88      !============================
89      ! READ SERIAL PORT
90      !============================
91      PROC readCom1()
92          VAR rawbytes raw_data;
93          VAR num length;
94          VAR string cadena;
95          ! ReadAnyBin SPIRIT.channel, answer \Time:=10;
96          ReadRawBytes SPIRIT.channel,raw_data\Time:=10;
97          length:=RawBytesLen(raw_data);
98          UnpackRawBytes raw_data,1,answer\ASCII:=length;
99          !   Sustitute LF=>% y CR=>&
100         cadena:=StrMap(answer,"\0A\0D","%&");
101         IF iprint THEN
102             TPWrite "<="+cadena+"("+NumToStr(length,0)+")";
103         ENDIF
104     ERROR
105         IF ERRNO=ERR_DEV_MAXTIME THEN
106             TPWrite "ERROR: Timeout";
107         ELSE
108             TPWrite "ERROR: de lectura";
109         ENDIF
110         RETURN ;
```

```
111     ENDPROC
112
113     !============================
114     ! WRITE SERIAL PORT
115     !============================
116     PROC writeCom1()
117         WriteStrBin SPIRIT.channel,query;
118         IF iprint THEN
119             TPWrite "=>"+query;
120         ENDIF
121     ERROR
122         TPWrite "ERROR: de escritura";
123         RETURN ;
124     ENDPROC
125
126     !============================
127     ! SALIDA PWM DEL ARDUINO
128     !============================
129     FUNC num arduinoPWM(VAR num canal_PWM,VAR num nivel_PWM)
130         query:="SPWM"+NumToStr(canal_PWM,0)+"\0A\0D";
131         writeCom1;
132         query:=NumToStr(nivel_PWM,0)+"\0A\0D";
133         writeCom1;
134         RETURN lee_arduinoPWM(canal_PWM,nivel_PWM);
135     ENDFUNC
136
137     !================================
138     ! ENTRADA ANALOGICA DEL ARDUINO
139     !================================
140     FUNC num arduinoAI(VAR num canal_AI)
141         query:="SRAI"+NumToStr(canal_AI,0)+"\0A\0D";
142         writeCom1;
143         RETURN lee_arduinoAI(canal_AI);
144     ENDFUNC
145
146     !============================
147     ! LEE PWM ARDUINO
148     !============================
149     FUNC num lee_arduinoPWM(VAR num canal_PWM,VAR num nivel_PWM)
150         VAR num lterm;
151         !   terminador
152         VAR num minNchan:=0;
153         ! valor  m ximo  del  n mero  de  canal
154         VAR num maxNchan:=5;
155         ! valor  m nimo  del  n mero  de  canal
156         VAR num minVchan:=0;
157         ! valor  m ximo  del  nivel  de  canal
158         VAR num maxVchan:=255;
159         ! valor  m nimo  del  nivel  de  canal
160         !
161         VAR bool iret;
162         VAR num largo;
163         VAR num p1:=0;
164         VAR num p2:=0;
165         VAR num p3:=0;
166         VAR string part;
167         VAR num canal;
168         VAR num nivel;
```

```
169         readCom1;
170         !answer:="SPWM0\0A\0D128\0A\0D";
171         lterm:=StrLen(SPIRIT.terminator);
172         largo:=StrLen(answer);
173         p1:=StrMatch(answer,1,"SPWM");
174         IF p1<largo THEN
175             p2:=StrFind(answer,p1+4,SPIRIT.terminator);
176             IF p2<largo THEN
177                 p3:=StrFind(answer,p2+lterm,SPIRIT.terminator);
178             ENDIF
179         ENDIF
180         IF (p1>0) AND (p2>p1+4) AND (p3>p2+lterm) AND (p3<largo) THEN
181             part:=StrPart(answer,p1+4,p2-(p1+4));
182             iret:=StrToVal(part,canal);
183             IF iret THEN
184                 canal:=Round(canal);
185                 IF ((canal>=minNchan) AND (canal<=maxNchan)) THEN
186                     IF canal=canal_PWM THEN
187                         part:=StrPart(answer,p2+lterm,p3-(p2+lterm));
188                         iret:=StrToVal(part,nivel);
189                         IF iret THEN
190                             nivel:=Round(nivel);
191                             IF ((nivel>=minVchan) AND (nivel<=maxVchan
                                    )) THEN
192                                 IF nivel=nivel_PWM THEN
193                                     RETURN nivel;
194                                 ELSE
195                                     TPWrite "PWM level = "+NumToStr(
                                            nivel,0)+
196                                     " no coincide con="+NumToStr(
                                            nivel_PWM,0);
197                                     RETURN -1;
198                                     ! Error, no coincide el nivel de
                                            canal
199                                 ENDIF
200                             ELSE
201                                 TPWrite "PWM level = "+NumToStr(nivel
                                        ,0);
202                                 RETURN -2;
203                                 ! Error en el nivel de canal (fuera de
                                        rango)
204                             ENDIF
205                         ELSE
206                             TPWrite "part="+part;
207                             RETURN -3;
208                             ! Error al obtener el nivel de canal
209                         ENDIF
210                     ELSE
211                         TPWrite "AI Channel = "+NumToStr(canal,0)+
212                         " no coincide con="+NumToStr(canal_PWM,0);
213                         RETURN -4;
214                         ! ERROR, no coincide el n mero de canal
215                     ENDIF
216                 ELSE
217                     TPWrite "AI Channel = "+NumToStr(canal,0);
218                     RETURN -5;
219                     ! Error en el n mero de canal (fuera de rango)
220                 ENDIF
```

```
221             ELSE
222                 TPWrite "part="+part;
223                 RETURN -6;
224                 ! Error al obtener el n mero de canal
225             ENDIF
226         ELSE
227             TPWrite "p1="+NumToStr(p1,0)+" p2="+NumToStr(p2,0)+" p3="+
                    NumToStr(p3,0);
228             RETURN -7;
229             ! Error en los punteros
230         ENDIF
231     ENDFUNC
232
233     !============================
234     ! LEE AI ARDUINO
235     !============================
236     FUNC num lee_arduinoAI(VAR num canal_AI)
237         VAR num lterm;
238         !   terminador
239         VAR num minNchan:=0;
240         ! valor  m ximo  del  n mero  de  canal
241         VAR num maxNchan:=5;
242         ! valor  m nimo  del  n mero  de  canal
243         VAR num minVchan:=0;
244         ! valor  m ximo  del  nivel  de  canal
245         VAR num maxVchan:=1023;
246         ! valor  m nimo  del  nivel  de  canal
247         !
248         VAR bool iret;
249         VAR num largo;
250         VAR num p1;
251         VAR num p2;
252         VAR num p3;
253         VAR string part;
254         VAR num canal;
255         VAR num nivel;
256         readCom1;
257         !answer:="SRAI0\0A\0D1008\0A\0D";
258         lterm:=StrLen(SPIRIT.terminator);
259         largo:=StrLen(answer);
260         p1:=StrMatch(answer,1,"SRAI");
261         IF p1<largo THEN
262             p2:=StrFind(answer,p1+4,SPIRIT.terminator);
263             IF p2<largo THEN
264                 p3:=StrFind(answer,p2+lterm,SPIRIT.terminator);
265             ENDIF
266         ENDIF
267         IF (p1>0) AND (p2>p1+4) AND (p3>p2+lterm) THEN
268             part:=StrPart(answer,p1+4,p2-(p1+4));
269             iret:=StrToVal(part,canal);
270             IF iret THEN
271                 canal:=Round(canal);
272                 IF ((canal>=minNchan) AND (canal<=maxNchan)) THEN
273                     IF canal=canal_AI THEN
274                         part:=StrPart(answer,p2+lterm,p3-(p2+lterm));
275                         iret:=StrToVal(part,nivel);
276                         IF iret THEN
277                             nivel:=Round(nivel);
```

```
278                             IF ((nivel>=minVchan) AND (nivel<=maxVchan
                                    )) THEN
279                                 RETURN nivel;
280                             ELSE
281                                 TPWrite "AI level = "+NumToStr(nivel
                                        ,0);
282                                 RETURN -1;
283                                 ! Error en el nivel de canal (fuera de
                                        rango)
284                             ENDIF
285                         ELSE
286                             TPWrite "part="+part;
287                             RETURN -2;
288                             ! Error al obtener el nivel de canal
289                         ENDIF
290                     ELSE
291                         TPWrite "AI Channel = "+NumToStr(canal,0)+
292                         " no coincide con="+NumToStr(canal_AI,0);
293                         RETURN -3;
294                         ! Error, no coincie el n mero de canal
295                     ENDIF
296                 ELSE
297                     TPWrite "AI Channel = "+NumToStr(canal,0);
298                     RETURN -4;
299                     ! Error en el n mero de canal (fuera de rango)
300                 ENDIF
301             ELSE
302                 TPWrite "part="+part;
303                 RETURN -5;
304                 ! Error al obtener el n mero de canal
305             ENDIF
306         ELSE
307             TPWrite "p1="+NumToStr(p1,0)+"p2="+NumToStr(p2,0)+"p3="+
                    NumToStr(p3,0);
308             RETURN -6;
309             ! Error en los punteros
310         ENDIF
311     ENDFUNC
312
313
314 ENDMODULE
```

Código 11.6: Módulo RAPID módulo generación de trayectorias (Track.mod)

```
%%%
    VERSION: 1
    LANGUAGE: ENGLISH
%%%

MODULE Track
    !
    !   Module for the realization of elementary trajectories
    !

    PROC rectangle(num long,num high,pos corner,speeddata vel,num acel
       )
        VAR robtarget p;
        VAR pos p_start;
        VAR pos p_end;
        !
        !  Horizontal Line
        p_start:=[corner.x,corner.y,corner.z];
        p_end:=[corner.x+long,corner.y,corner.z];
        markLine p_start,p_end,vel,acel;
        !
        !  Vertical Line
        p_start:=[corner.x+long,corner.y,corner.z];
        p_end:=[corner.x+long,corner.y,corner.z-long];
        markLine p_start,p_end,vel,acel;
        !
        !  Horizontal Line
        p_start:=[corner.x+long,corner.y,corner.z-long];
        p_end:=[corner.x,corner.y,corner.z-long];
        markLine p_start,p_end,vel,acel;
        !
        !  Vertical Line
        p_start:=[corner.x,corner.y,corner.z-long];
        p_end:=[corner.x,corner.y,corner.z];
        markLine p_start,p_end,vel,acel;
        !
    ENDPROC

    PROC cross(num long,pos corner,speeddata vel,num acel)
        VAR robtarget p;
        VAR pos p_start;
        VAR pos p_end;
        !
        !  Horizontal Line
        p_start:=[corner.x-long/4,corner.y,corner.z];
        p_end:=[corner.x+3*long/4,corner.y,corner.z];
        markLine p_start,p_end,vel,acel;
        !
        !  Vertical Line
        p_start:=[corner.x,corner.y,corner.z+long/4];
        p_end:=[corner.x,corner.y,corner.z-3*long/4];
        markLine p_start,p_end,vel,acel;
        !
    ENDPROC

```

```
55      PROC linesDy(num long,num high,num deep,num nlin,pos corner,
            speeddata vel,num acel)
56          VAR robtarget p;
57          VAR pos p_start;
58          VAR pos p_end;
59          !
60          !       Lines in the XZ plane with different Y
61          FOR k FROM 1 TO nlin DO
62              p_start:=[corner.x-long/2,corner.y+deep*(k-1)/(nlin-1),
                    corner.z-high*(k-1)/(nlin-1)];
63              p_end:=[corner.x+long/2,corner.y+deep*(k-1)/(nlin-1),
                    corner.z-high*(k-1)/(nlin-1)];
64              markLine p_start,p_end,vel,acel;
65          ENDFOR
66      ENDPROC
67
68      PROC markLine(pos p_start,pos p_end,speeddata velocidad,num acel)
69          VAR robtarget p;
70          VAR num time;
71          VAR num lane;
72          VAR pos direc;
73          VAR num long;
74          !
75          ! time for acceleration lane in seconds
76          time:=velocidad.v_tcp/acel;
77          ! Acceleration lane in mm
78          lane:=Pow(velocidad.v_tcp,2)/2/acel;
79          direc:=p_end-p_start;
80          long:=Sqrt(Pow(direc.x,2)+Pow(direc.y,2)+Pow(direc.z,2));
81          direc:=direc*(1/long);
82          p_start:=p_start-lane*direc;
83          p_end:=p_end+0*lane*direc;
84          !
85          !   Start of line
86          p:=[p_start,quat,cuadrantes,ejexT];
87          MoveL p,velocidad,fine,tCenefaF3;
88          !WaitUntil \Inpos, TRUE;
89          !
90          SetDO\SDelay:=time,laser_DO,1;
91          !
92          !   Marking the line
93          p:=[p_end,quat,cuadrantes,ejexT];
94          MoveL p,velocidad,fine,tCenefaF3;
95          !
96          SetDO\SDelay:=0,laser_DO,0;
97      ENDPROC
98
99  ENDMODULE
```

Código 11.7: Módulo RAPID programa ejemplo (MainModule.mod)

```
%%%
    VERSION: 1
    LANGUAGE: ENGLISH
%%%

MODULE MainModule
    VAR signaldo laser_DO;

    TASK PERS tooldata tCenefaF3:=[TRUE
       ,[[-4.10,-35.30,32.40],[0.707107,0.000000,-0.707107,0.000000]],[0.400,[-
       e-04,4.00e-04,4.86e-04]];
    VAR orient quat;
    VAR confdata cuadrantes;
    CONST extjoint ejexT:=[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09];
    CONST jointtarget jpos0:=[[0,0,0,0,0,0],ejexT];
    CONST jointtarget jposAvia:=[[0,0,0,90,90,-90],ejexT];

    PROC main()
        !
        !   initialize
        init;
        !
        ! AVIA
        select_laser("AVIA");
        !
        ! test_tcp_AVIA;  ! Test TPC/IP communication with AVIA
        !         avia_lines; ! AVIA lines
                avia_linesF; ! AVIA lines freq
        ! avia_cross; ! AVIA cross
        !         avia_rectangles; ! AVIA rectangles
        !
        ! SPIRIT
        select_laser "SPIRIT";
        !
        !test_arduino; ! Arduino read/write test
        !spirit_steps; ! Spirit Power Steps
        !
        !
    ENDPROC

    PROC init()
        !   Avia GATE on Digital_Output_1, OFF
        SetDO D652_10_DO1,0;
        !   Spirit GATE on Digital_Output_2, OFF
        SetDO D652_10_DO2,0;
    ENDPROC

    PROC select_laser(string laser_name)
        IF laser_name="AVIA" THEN
            !
            !   Avia GATE on Digital_Output_1
            AliasIO D652_10_DO1,laser_DO;
            TPWrite "Laser_DO = D652_10_DO1";
        ELSEIF laser_name="SPIRIT" THEN
            !
```

```
54              !   Spirit GATE on Digital_Output_2
55              AliasIO D652_10_DO2,laser_DO;
56              TPWrite "Laser_DO = D652_10_DO2";
57          ELSE
58              TPWrite "unknown Laser";
59          ENDIF
60      ENDPROC
61
62      PROC test_tcp_AVIA()
63          VAR bool result;
64          ! Create and connect the socket
65          avia_open;
66          ! Test response AVIA
67          avia_test;
68          !   Open shutter
69          result:=avia_diodes_on();
70          result:=avia_set_pulse_mode(0);
71          result:=avia_pulsing_on();
72          result:=avia_shutter_open();
73          !   Set laser frequency
74          result:=avia_set_frequency(12.345E03);
75          !   Set Diode Current
76          result:=avia_set_current(78.9);
77          !   Wait 1 second
78          WaitTime 1;
79          !   Close shutter
80          result:=avia_shutter_close();
81          result:=avia_pulsing_off();
82          result:=avia_diodes_off();
83          ! Close the socket
84          avia_close;
85      ENDPROC
86
87      PROC AviaSetUp(num frequency,num current)
88          VAR bool result;
89          ! Create and connect the socket
90          avia_open;
91          ! Test response AVIA
92          avia_test;
93          !   Open shutter
94          !!          result:=avia_diodes_on();
95          !!          result:=avia_set_pulse_mode(0);
96          !!          result:=avia_pulsing_on();
97          result:=avia_shutter_open();
98          !   Set laser frequency
99          result:=avia_set_frequency(frequency);
100         !   Set diode current
101         result:=avia_set_current(current);
102     ENDPROC
103
104
105     PROC AviaOff()
106         VAR bool result;
107         !   Close shutter
108         result:=avia_shutter_close();
109         !!          result:=avia_pulsing_off();
110         !!          result:=avia_diodes_off();
111         ! Close the socket
```

```
112         avia_close;
113     ENDPROC
114
115     PROC avia_lines()
116         VAR robtarget p;
117         VAR pos center:=[530,100,405];
118         VAR pos position;
119         VAR num long;
120         VAR num high;
121         VAR num deep;
122         VAR num nlin;
123         VAR speeddata vel;
124         VAR num acel;
125         !
126         !   Seting up the laser: frequency, currente
127         AviaSetUp 1000,90.0;
128         !
129         !   Rest position
130         !   MoveAbsJ jpos20\NoEOffs,v1000,fine,tCenefaF3;
131         !
132         !   Move to the position with tool orientation
133         position:=[530,100,405];
134         quat:=[0.72469,0.00748,-0.05663,0.68670];
135         cuadrantes:=[0,1,-2,0];
136         p:=[center,quat,cuadrantes,ejexT];
137         MoveL p,v100,fine,tCenefaF3;
138         !
139         !   Wait for confirmation
140         UIMsgBox "Continue the program ?";
141         !
142         !   Do the lines
143         position:=[545,100,380];
144         long:=10;
145         high:=10;
146         deep:=5;
147         nlin:=11;
148         vel:=v5;
149         acel:=10;
150         linesDy long,high,deep,nlin,position,vel,acel;
151         !
152         !   Laser OFF
153         AviaOff;
154         !
155         !   Move the object out of the laser beam
156         position:=[530,100,455];
157         p:=[position,quat,cuadrantes,ejexT];
158         MoveL p,v100,fine,tCenefaF3;
159         !
160         !   Rest position
161         !       MoveAbsJ jpos20\NoEOffs,v1000,fine,tCenefaF3;
162         !
163         !   HOME position
164         !       MoveAbsJ jpos0\NoEOffs,v1000,fine,tCenefaF3;
165     ENDPROC
166
167     PROC avia_linesF()
168         VAR robtarget p;
169         VAR pos center:=[530,100,405];
```

```
170         VAR pos position;
171         VAR num long;
172         VAR num high;
173         VAR num deep;
174         VAR num nlin;
175         VAR speeddata vel;
176         VAR num acel;
177         VAR pos p_start;
178         VAR pos p_end;
179         VAR num f_start;
180         VAR num f_end;
181         VAR num freq;
182         VAR bool result;
183         !
184         !   Rest position
185         !   MoveAbsJ jpos20\NoEOffs,v1000,fine,tCenefaF3;
186         !
187         !   Seting up the laser: frequency, currente
188             AviaSetUp 1000,90.0;
189         !
190         !   Move to the position with tool orientation
191         position:=[530,100,405];
192         quat:=[0.72469,0.00748,-0.05663,0.68670];
193         cuadrantes:=[0,1,-2,0];
194         p:=[center,quat,cuadrantes,ejexT];
195         MoveL p,v100,fine,tCenefaF3;
196         !
197         !   Wait for confirmation
198         UIMsgBox "Continue the program ?";
199         !
200         !   Do the lines
201         position:=[545,101,380];
202         long:=10;
203         high:=5;
204         f_start := 5;
205         f_end := 50;
206         nlin:=10;
207         vel:=v5;
208         acel:=10;
209         FOR k FROM 1 TO nlin DO
210         !
211         !   Seting up the laser: frequency
212             freq := f_start + (f_end - f_start)*(k-1)/(nlin-1);
213             result:=avia_set_frequency(freq);
214             WaitTime 0.5;
215             p_start:=[position.x-long/2,position.y,position.z-high*(k
                    -1)/(nlin-1)];
216             p_end:=[position.x+long/2,position.y,position.z-high*(k-1)
                    /(nlin-1)];
217             markLine p_start,p_end,vel,acel;
218         ENDFOR
219         !
220         !   Laser OFF
221         AviaOff;
222         !
223         !   Move the object out of the laser beam
224         position:=[530,100,455];
225         p:=[position,quat,cuadrantes,ejexT];
```

```
226         MoveL p,v100,fine,tCenefaF3;
227         !
228         !   Rest position
229         !       MoveAbsJ jpos20\NoEOffs,v1000,fine,tCenefaF3;
230         !
231         !   HOME position
232         !       MoveAbsJ jpos0\NoEOffs,v1000,fine,tCenefaF3;
233     ENDPROC
234
235     PROC avia_cross()
236         !       VAR string receive_string;
237         VAR robtarget p;
238         VAR pos center:=[530,100,405];
239         VAR pos position;
240         !
241         !   Seting up the laser: frequency, currente
242         AviaSetUp 1000,90.0;
243         !
244         !   Rest position
245         !   MoveAbsJ jpos20\NoEOffs,v1000,fine,tCenefaF3;
246         !
247         !   Move to the position with tool orientation
248         position:=[530,101,390];
249         quat:=[0.72469,0.00748,-0.05663,0.68670];
250         cuadrantes:=[0,1,-2,0];
251         p:=[center,quat,cuadrantes,ejexT];
252         MoveL p,v100,fine,tCenefaF3;
253         !
254         !   Wait for confirmation
255         UIMsgBox "Continue the program ?";
256         !
257         !   Do the cross: long, corner, vel, acel
258         cross 10,[545,101,390],v5,10;
259         !
260         !   Laser OFF
261         AviaOff;
262         !
263         !   Move the object out of the laser beam
264         position:=[530,80,455];
265         p:=[position,quat,cuadrantes,ejexT];
266         MoveL p,v100,fine,tCenefaF3;
267         !
268         !   Rest position
269         !       MoveAbsJ jpos20\NoEOffs,v1000,fine,tCenefaF3;
270         !
271         !   HOME position
272         !       MoveAbsJ jpos0\NoEOffs,v1000,fine,tCenefaF3;
273     ENDPROC
274
275     PROC avia_rectangles()
276         !       VAR string receive_string;
277         VAR robtarget p;
278         VAR pos center:=[530,100,405];
279         VAR pos position;
280         !
281         !   Seting up the laser: frequency, currente
282         AviaSetUp 1000,90.0;
283         !
```

```
284         !   Rest position
285         !   MoveAbsJ jpos20\NoEOffs,v1000,fine,tCenefaF3;
286         !
287         !   Move to the position with tool orientation
288         position:=[530,100,405];
289         quat:=[0.72469,0.00748,-0.05663,0.68670];
290         cuadrantes:=[0,1,-2,0];
291         p:=[center,quat,cuadrantes,ejexT];
292         MoveL p,v100,fine,tCenefaF3;
293         !
294         !   Wait for confirmation
295         UIMsgBox "Continue the program ?";
296         !         !
297         !         !   Do the cross
298         !         cross 10,[545,103.5,405],v5;
299         !         !
300         !         !   Do the lines
301         !         centro:=[545+3, 100, 405-3];
302         !         largo:=10;
303         !         alto:=10;
304         !         profundo:=5;
305         !         nlin:=11;
306         !         velocidad:=v5;
307         !         linesDy largo,alto,profundo,nlin, centro, velocidad;
308         !
309         !   Do the rectangles
310         rectangle 10,10,[545+15*0,103.5,400-15*0],v5,60;
311         rectangle 10,10,[545+15*1,103.5,400-15*0],v5,50;
312         rectangle 10,10,[545+15*2,103.5,400-15*0],v5,40;
313         rectangle 10,10,[545+15*0,103.5,400-15*1],v5,30;
314         rectangle 10,10,[545+15*1,103.5,400-15*1],v5,20;
315         rectangle 10,10,[545+15*2,103.5,400-15*1],v5,10;
316         !
317         !   Laser OFF
318         AviaOff;
319         !
320         !   Move the object out of the laser beam
321         position:=[530,100,455];
322         p:=[position,quat,cuadrantes,ejexT];
323         MoveL p,v100,fine,tCenefaF3;
324         !
325         !   Rest position
326         !         MoveAbsJ jpos20\NoEOffs,v1000,fine,tCenefaF3;
327         !
328         !   HOME position
329         !         MoveAbsJ jpos0\NoEOffs,v1000,fine,tCenefaF3;
330     ENDPROC
331
332     !============================
333     ! ARDUINO TEST
334     !============================
335     PROC test_arduino()
336         VAR num nivel_PWM:=128;
337         VAR num canal_PWM:=0;
338         VAR num canal_AI:=0;
339         VAR num resp_nivel_AI;
340         VAR num resp_nivel_PWM;
341         VAR num tension;
```

```
342         !
343         TPWrite "INICIO: "+CDate()+CTime();
344         IF iprint THEN
345             TPWrite "iprint=TRUE";
346         ELSE
347             TPWrite "iprint=FALSE";
348         ENDIF
349         !
350         !   ABRE COM1
351         spirit_open;
352         !
353         !   ARDUINO_PWM
354 !        TPReadNum tension,"Esperando un n mero (0.0 a 5.0)";
355 !        nivel_PWM:=trunc(255*tension/5);
356         !
357         TPReadNum nivel_PWM,"Esperando un n mero (0 a 255)";
358         nivel_PWM:=trunc(nivel_PWM);
359         !
360         resp_nivel_PWM:=arduinoPWM(canal_PWM,nivel_PWM);
361         IF resp_nivel_PWM>=0 THEN
362             TPWrite "PWM Level = "+NumToStr(resp_nivel_PWM,0);
363         ELSE
364             TPWrite "ERROR: leyendo PWM, code = "+NumToStr(
                    resp_nivel_PWM,0);
365         ENDIF
366         !   ARDUINO_AI
367         resp_nivel_AI:=arduinoAI(canal_AI);
368         IF resp_nivel_AI>=0 THEN
369             TPWrite "AI Level = "+NumToStr(resp_nivel_AI,0);
370         ELSE
371             TPWrite "ERROR: leyendo AI, code = "+NumToStr(
                    resp_nivel_AI,0);
372         ENDIF
373         !
374         !   CIERRA COM1
375         spirit_close;
376         !
377     ENDPROC
378
379     !============================
380     ! SPIRIT STEPS
381     !============================
382     PROC spirit_steps()
383         VAR num power;
384         VAR bool result;
385         !
386         TPWrite "INICIO: "+CDate()+CTime();
387         IF iprint THEN
388             TPWrite "iprint=TRUE";
389         ELSE
390             TPWrite "iprint=FALSE";
391         ENDIF
392         !
393         !   ABRE COM1
394         spirit_open;
395         !
396         !   Wait for confirmation
397         UIMsgBox "Continue the program ?";
```

```
398         !
399         !   LAZO
400         FOR power FROM 100 TO 10 STEP -10 DO
401             result:=spirit_set_power(power);
402             WaitTime 0.2;
403             SetDO\SDelay:=0,laser_DO,1;
404             WaitTime 0.2;
405             SetDO\SDelay:=0,laser_DO,0;
406             !            PulseDO \PLength:=0.010, Laser_DO;
407         ENDFOR
408         ! fin lazo
409         !
410         !   CIERRA COM1
411         spirit_close;
412         !
413     ENDPROC
414
415 ENDMODULE
```

Código 11.8: Módulo RAPID para importar los módulos (LasersProgram.pgf)

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<Program>
    <Module>Track.mod</Module>
    <Module>SpiritArduino.mod</Module>
    <Module>MainModule.mod</Module>
    <Module>AviaTcp.mod</Module>
</Program>
```

12 CÁLCULOS

En este apartado se exponen los cálculos llevados a cabo para el dimensionamiento de los componentes electrónicos en los distintos circuitos.

12.1. Optoacoplador AVIA

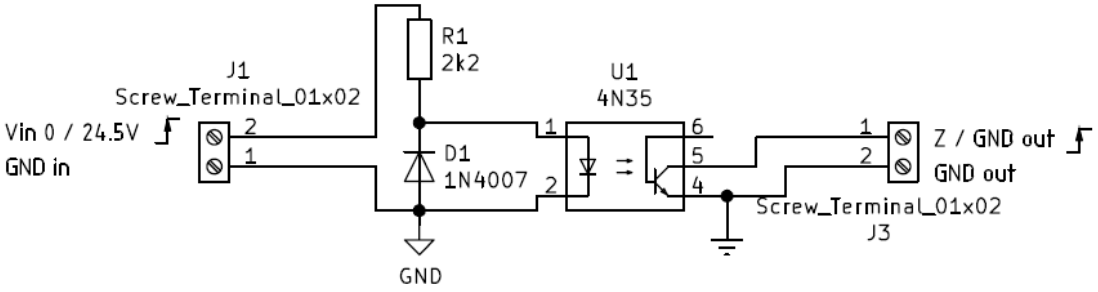


Figura 12.1 – Circuito optoacoplador láser AVIA

En el circuito de arriba el único componente a dimensionar es la resistencia R1. Para ello se parte de los siguientes datos:

- Tensión de entrada a nivel alto: $Vin = 24V$
- Tensión de polarización del LED del optoacoplador: $V_{LED} = 1,7V$
- Corriente a través del LED del optoacoplador: $I_{LED} = 10mA$

De los que se deduce:

$$24V = V_R + V_{LED} \quad (12.1)$$
$$= I_{LED}R1 + V_{LED} \quad (12.2)$$
$$(12.3)$$

Despejando:

$$R1 = \frac{24V - V_{LED}}{I_{LED}} \quad (12.4)$$
$$= \frac{24V - 1,7V}{10mA} \quad (12.5)$$
$$= 2,23k\Omega \quad (12.6)$$
$$\mathbf{R1 = 2k2} \quad (12.7)$$
$$(12.8)$$

12.2. Optoacoplador Spirit

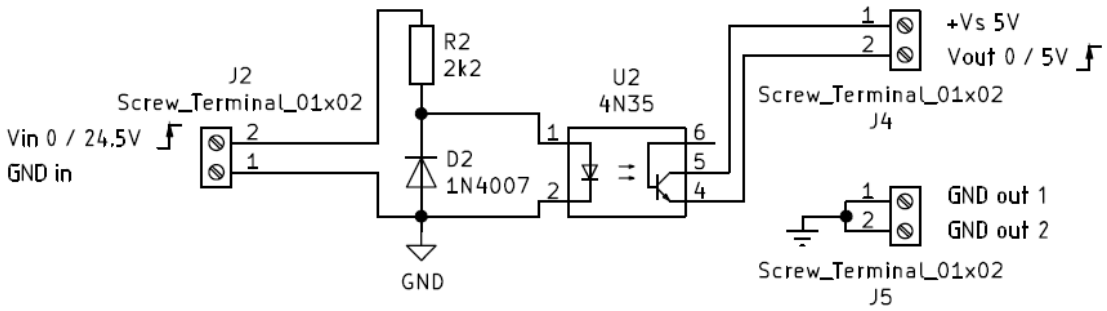


Figura 12.2 – Circuito optoacoplador láser Spirit

En el circuito de arriba el único componente a dimensionar es la resistencia R2. Como esta mitad del circuito es igual a la del otro optoacoplador se dimensiona de la misma manera y tiene el mismo valor.

R2=2k2 (12.9)

12.3. DAQ Arduino

El hardware de la DAQ Arduino aparace representado en el siguiente diagrama:

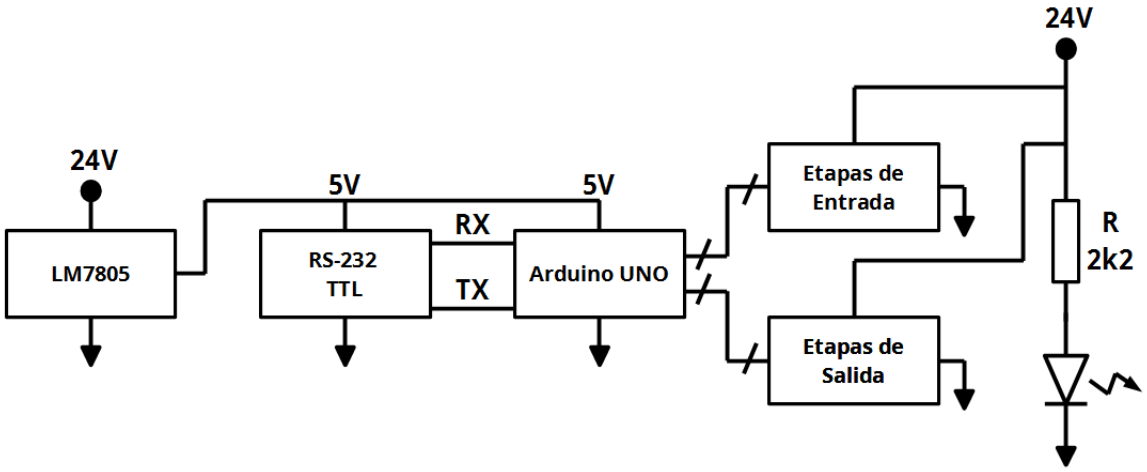


Figura 12.3 – Diagrama DAQ Arduino

Para realizar los cálculos se separó el problema en las siguientes partes:

1. Acondicionamiento del regulador de tensión lineal LM7805.
2. Programación Arduino UNO R3.
3. Etapas de entrada.
4. Etapas de salida.
5. Resistencia y LED de encendido.

12.3.1. Acondicionamiento LM7805

Este regulador lineal de tensión sirve para alimentar el módulo RS-232 a TTL y el Arduno UNO. Para ello se ha de obtener $5V_{DC}$ a partir de los $24,5V_{DC}$ de la fuene de alimentación del brazo robot.

Para acondicionar el regulador se siguieron las recomendaciones del datasheet del fabricante. En el se indicaba que simplemente era necesario colocar un condensador de $0,33\mu F$ entre las patillas de $24,5V$ y GND y otro de $0,1\mu F$ entre la salida de $5V$ y GND común.

12.3.2. Programación Arduino UNO R3

Este punto es crucial para los cálculos posteriores de las etapas de salida.

Por defecto en el entorno de desarrollo Arduino IDE las salidas analógicas PWM de la placa se generan empleando el comando *analogWrite(pin,value)*. Para muchos casos este

comando cumple con los requerimientos, pero este no es el caso ya que no permite controlar la frecuencia de las señales de salida PWM.

Las salidas analógicas PWM se generan a través de los timers integrados en el microcontrolador del Arduino (ATMega328p). Estos timmers se controlan a través de una serie de registros de la memoria del microcontrolador reservados para tal efecto. El valor de estos registros se puede programar directamente a través de Arduino IDE.

En conclusión a la hora de programar el Arduino la generación de salidas analógicas se ha de hacer modificando el valor de registros del microcontrolador. Y esto es así, porque de este modo se pueden configurar todas las salidas PWM para que trabajen a la frecuencia máxima común. A su vez esto se justifica porque cuanto más alta sea la frecuencia de conmutación más sencillo será filtrar las salidas para obtener en las etapas de salida el valor de tensión continua amplificado que se desee (porque mayor frecuencia tendrán los armónicos que componen la señal cuadrada).

En el manual del microcontrolador ATMega328p se indica que los timers se pueden configurar en distintos modos de los cuales sólo dos sirven para generar una señal PWM. Estos son:

- **Fast PWM:** En este modo el timmer cuenta hasta el desborde donde conmuta la señal pasando por el punto de compare match donde también conmuta la señal.
- **Phase correct:** Funciona de manera muy similar, la diferencia es que cuando llega al desborde hace la cuenta hacia atrás desde el desborde al cero.

El modo más adecuado para la aplicación es Fast PWM porque permite unas frecuencias de PWM más elevadas aunque tenga una menor precisión.

En el modo escogido se genera una señal PWM de $8bits$ siendo la frecuencia de CLK del microcontrolador y el timer de $16MHz$. Por tanto la frecuencia de la señal PWM:

$$f_{PWM} = \frac{16MHz}{2^8} = 62,5KHz \tag{12.10}$$

A continuación se detalla la configuración de los timmers:

- **Timer 0**

Registro	Valor
TCCR0A	0b1010xx11
TCCR0B	0b00xx0001

Tabla 12.1 – Configuración Timer 0

Registro	Valor	PIN Arduino
OCR0A	0 a 255	6
OCR0B	0 a 255	5

Tabla 12.2 – Manejo de Timer 0

- **Timer 1**

Registro	Valor
TCCR1A	0b1010xx01
TCCR1B	0b00x01001
TCCR1C	0b00xxxxxx

Tabla 12.3 – Configuración Timer 1

Registro	Valor	PIN Arduino
OCR1A	0 a 255	9
OCR1B	0 a 255	10

Tabla 12.4 – Manejo de Timer 1

- **Timer 2**

Registro	Valor
TCCR2A	0b1010xx11
TCCR2B	0b00xx0001

Tabla 12.5 – Configuración Timer 2

Registro	Valor	PIN Arduino
OCR2A	0 a 255	11
OCR2B	0 a 255	3

Tabla 12.6 – Manejo de Timer 2

12.3.3. Etapas de entrada

Los esquemas de abajo representan la totalidad de la topología de las etapas de entrada. Véase que siempre se coloca una resistencia de $470k$ entre la entrada y masa. La razón de esto es asegurar que siempre se polarice de forma correcta el AO (Amplificador Operacional).

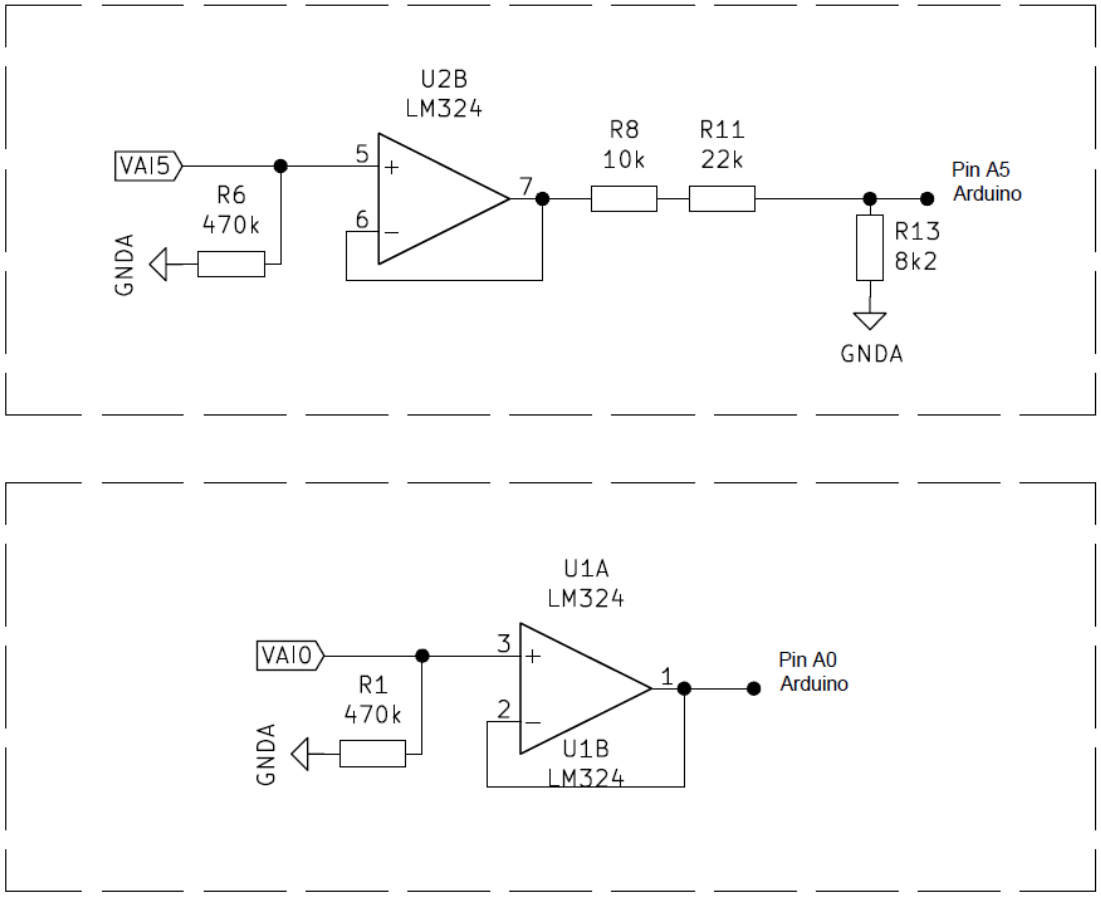


Figura 12.4 – DAQ Arduino circuito etapas de entrada

Se dimensionaron las siguientes entradas:

- **4 entradas analógicas V_{FS} = 0 a 5V** Corresponde con el segundo esquema de la imagen.

$$\frac{Vo}{Vi} = \frac{(5-0)V}{(5-0)V} = 1 \tag{12.11}$$

- **1 entrada analógica V_{FS} = 0 a 12V** Corresponde con el primer esquema de la imagen. Es necesario añadir un divisor de tensión.

$$\frac{Vo}{Vi} = \frac{(5-0)V}{(12-0)V} = 0,04167 \tag{12.12}$$

El divisor equivalente con valores normalizados:

$$\frac{8,2k}{8,2k + (10+1,5)k} = 0,4162 \tag{12.13}$$

Respecto a la imagen:

$$R8 = 10k \tag{12.14}$$
$$R11 = 1k5 \tag{12.15}$$
$$R13 = 8k2 \tag{12.16}$$
$$\tag{12.17}$$

- **1 entrada analógica V_{FS} = 0 a 24,5V** Corresponde con el primer esquema de la imagen. Es necesario añadir un divisor de tensión.

$$\frac{Vo}{Vi} = \frac{(5-0)V}{(24,5-0)V} = 0,2041 \tag{12.18}$$

El divisor equivalente con valores normalizados:

$$\frac{8,2k}{8,2k + (10+22)k} = 0,2039 \tag{12.19}$$

Respecto a la imagen:

$$R8 = 10k \tag{12.20}$$
$$R11 = 22k \tag{12.21}$$
$$R13 = 8k2 \tag{12.22}$$
$$\tag{12.23}$$

Esta salida satura en 23V aproximadamente, pero es algo esperado en el diseño de la plca ya que los aomplificadores operacionales no son rail to rail (Análogamente ocurre lo mismo en la salida AO6). La siguiente imagen corresponde al resultado de la simulación de la evolución de la tensión en la salida frente a la entrada de la etapa:

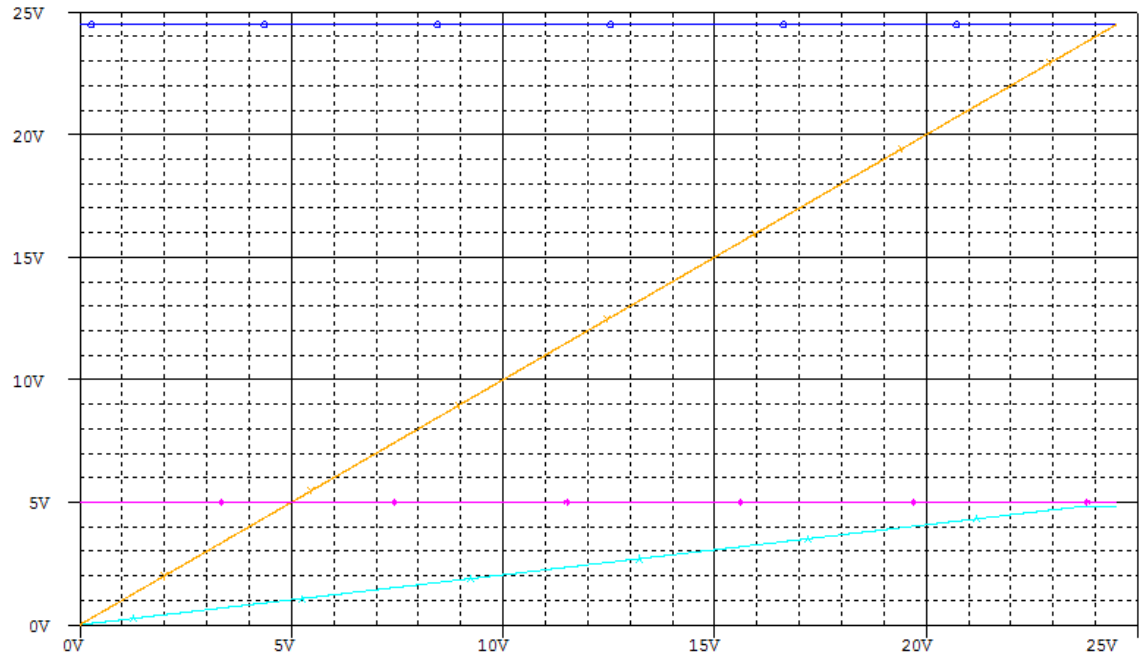


Figura 12.5 – DAQ Arduino AI5

12.3.4. Etapas de salida

Todas las etapas de salida tienen en común que el filtro a la salida del Arduino es para todas igual. Se trata de un filtro pasivo de primer orden de tipo RC.

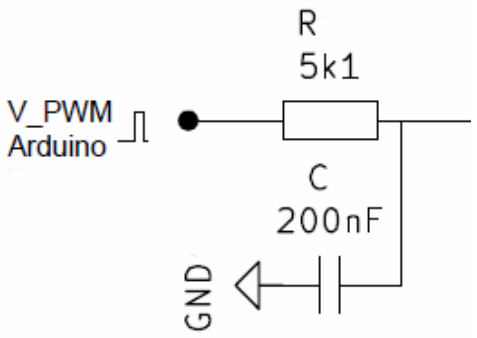


Figura 12.6 – Filtro RC

Para el dimensionamiento del filtro en un primer cálculo se idealizó la salida PWM del Arduino a una señal senoidal del doble de amplitud. De este modo se puede obtener un primer valor de los componentes para simular el circuito.

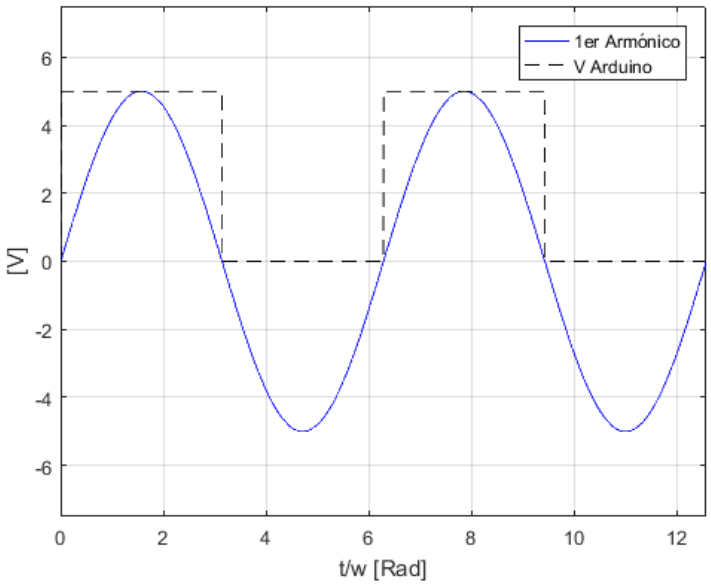


Figura 12.7 – Idealización cálculo

Entonces, se parte de los siguientes datos:

- Frecuencia: $F_{PWM} = 62KHz$
- Primer Armónico; $A = 5V$
- Rizado máximo admisible: $V_{RIZADO} <= \frac{1V}{80} = 12,5mV$

Calculando la función de transferencia:

$$G(s) = \frac{Vo}{Vin} \tag{12.24}$$

$$= \frac{1}{RCs+1} \tag{12.25}$$

Calculando el módulo ($s = j\omega$)($\omega = 2\pi f$:

$$|G(f)| = \frac{\sqrt{(1)^2}}{\sqrt{(RC2\pi f)^2 + (1)^2}} \tag{12.26}$$

$$= \frac{1}{\sqrt{(RC2\pi f)^2 + 1}} \tag{12.27}$$

Evaluando en $|G(f_{PWM}|$:

$$|G(f_{PWM})| = \frac{1}{\sqrt{(RC2\pi 62,5KHz)^2 + 1}} \tag{12.28}$$

$$|G(f_{PWM})| = \frac{V_{RIZADO}}{A} = \frac{12,5mV}{5V} \tag{12.29}$$

Despejando:

$$RC = 1,0268(10^{-3}) \tag{12.30}$$

Valores normalizados:

$$R = 200nF \tag{12.31}$$
$$C = 5K1 \tag{12.32}$$

Para calcular el tiempo de respuesta del filtro con estos valores, es decir, hacer su análisis transitorio partimos de la ecuación del condensador:

$$Vc = \int_0^t i_c(t)dt \tag{12.33}$$
$$\tag{12.34}$$

En forma diferencial y aplicado al filtro se tienen las ecuaciones:

$$i_c(t) = C\ \frac{dVc}{dt} \tag{12.35}$$
$$Vc = V_F - R\ ic(t) \tag{12.36}$$
$$\tag{12.37}$$

Operando:

$$Vc = V_F - RC\frac{dVc}{dt} \tag{12.38}$$
$$V_F - Vc = RC\frac{dVc}{dt} \tag{12.39}$$
$$\int_{Vo}^{Vc} \frac{1}{V_F - Vc}dVc = \int_0^t \frac{1}{RC}dt \tag{12.40}$$
$$-ln(V_F - Vc) + ln(V_F - V_I) = \frac{1}{RC}\Delta T \tag{12.41}$$
$$ln\left(\frac{V_F - V_I}{V_F - Vc}\right) = e^{\frac{\Delta T}{RC}} \tag{12.42}$$
$$\frac{V_F - V_I}{V_F - Vc} = \frac{\Delta T}{RC} \tag{12.43}$$
$$Vc = V_F + (V_I - V_F)e^{\frac{-\Delta T}{RC}} \tag{12.44}$$
$$Vc = V_F + (V_I - V_F)e^{\frac{-t}{\tau}}\ ;\ \tau = RC \tag{12.45}$$

Evaluando en $t = 5\tau$

$$
\begin{aligned}
t &= 5\tau && (12.46)\\
&= (5k1)(200nF) && (12.47)\\
&= 5,1ms && (12.48)\\
Vc(t = 5\tau) &= V_F + (V_I - V_F)\, e^{\frac{-t}{\tau}} && (12.49)\\
&= 5V + (0 - 5V)e^{-\frac{5\tau}{\tau}} && (12.50)\\
&= 4,966 && (12.51)\\
\frac{4,966V}{5V} &= 0,9932 && (12.52)\\
&= 99,32\,\% && (12.53)\\
& && (12.54)
\end{aligned}
$$

El tiempo de respuesta máximo del filtro (de 0 a $5V$) es de $5,1ms$.

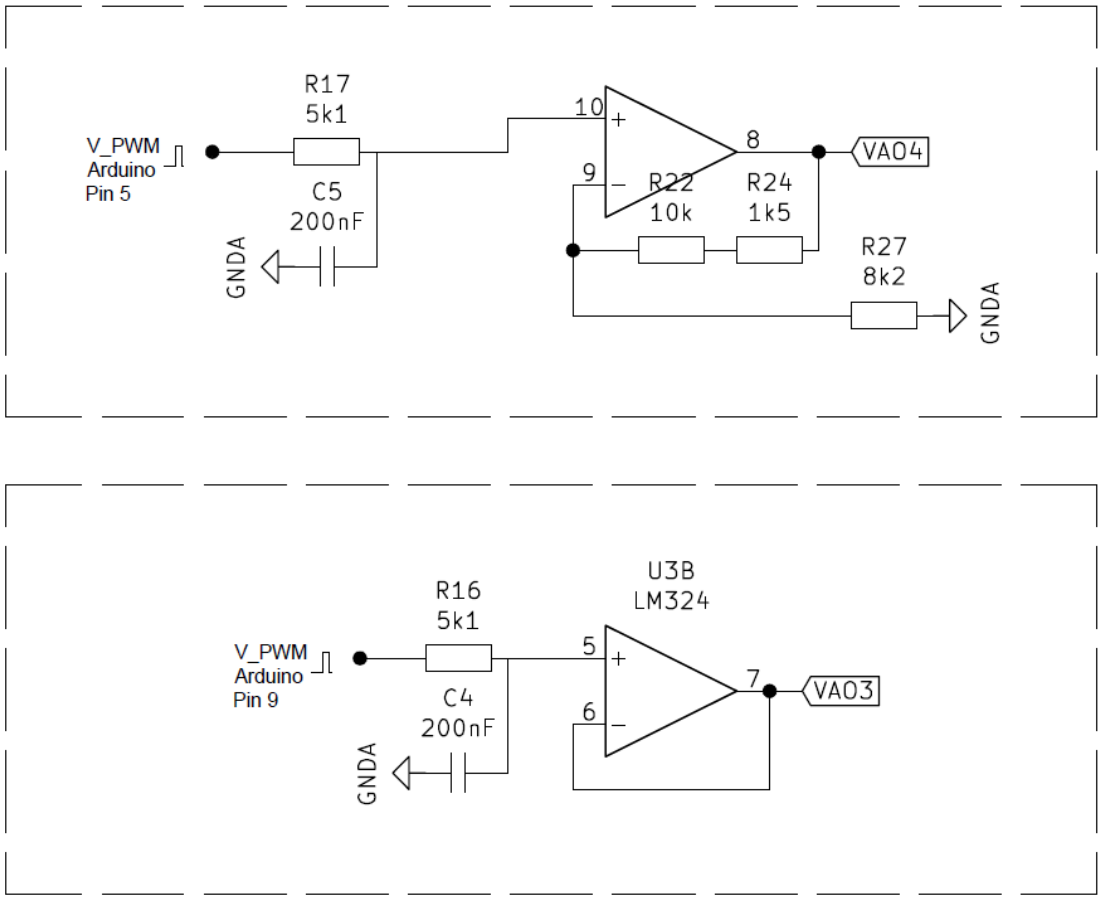


Figura 12.8 – DAQ Arduino circuito etapas de salida

De manera análoga a las entradas se dimensionaron las siguientes salidas:

- **4 salidas analógicas V_{FS} = 0 a 5V** Corresponde con el segundo esquema de la imagen.

$$\frac{Vo}{Vi} = \frac{(5-0)V}{(5-0)V} = 1 \tag{12.55}$$

En las siguientes gráficas se puede observar la respuesta simulada de la etapa para unas salidas de 2,5V y 5V respectivamente. El tiempo de respuesta y el rizado son los esperados y el amplificador no influye en la respuesta:

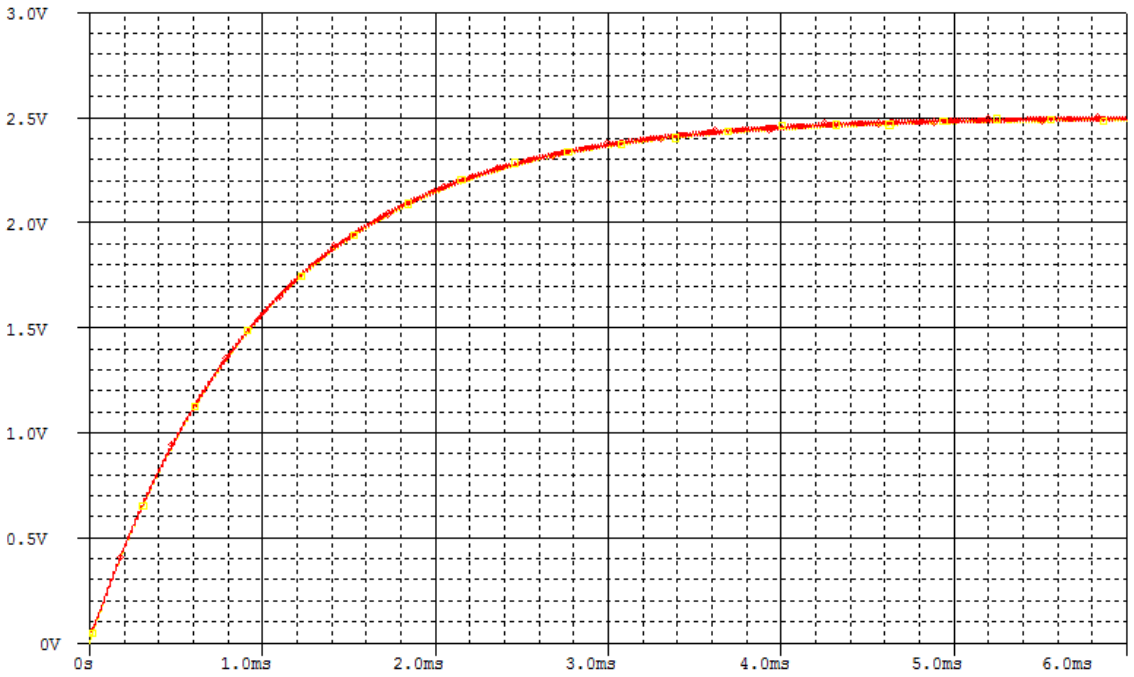


Figura 12.9 – DAQ Arduino AO0 (0 a 2,5V)

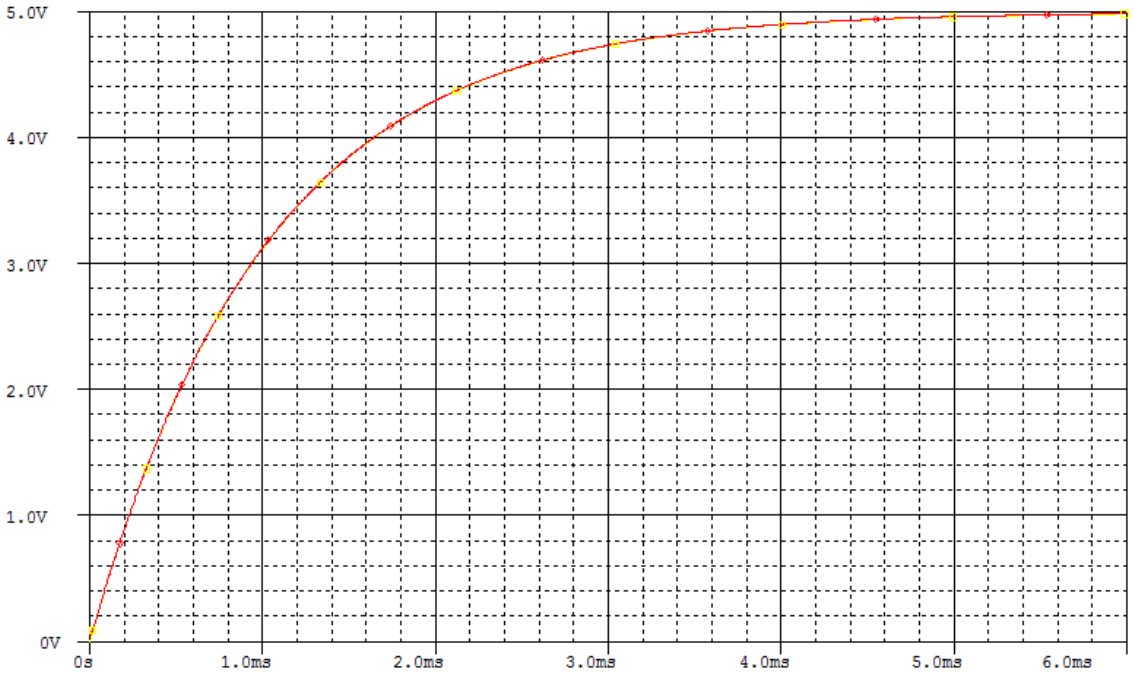


Figura 12.10 – DAQ Arduino AO0 (0 a 5V)

- **1 salida analógica V_{FS} = 0 a 12V** Corresponde con el primer esquema de la imagen. Es necesario modificar la ganancia del montaje mediante resistencias.

$$G = \frac{(12-0)V}{(5-0)V} = 2,4 \tag{12.56}$$

La ganancia equivalente con valores normalizados de resistencias:

$$G = \left(1 + \frac{10k + 1,5k}{8,2k}\right) = 2,402 \tag{12.57}$$

Respecto a la imagen:

$$R22 = 10k \tag{12.58}$$
$$R24 = 1k5 \tag{12.59}$$
$$R27 = 8k2 \tag{12.60}$$
$$\tag{12.61}$$

- **1 entrada analógica V_{FS} = 0 a 24,5V** Corresponde con el primer esquema de la imagen. Es necesario añadir modificar la ganancia del amplificador mediante resistencias.

$$G = \frac{(24,5-0)V}{(5-0)V} = 4,9 \tag{12.62}$$

La ganancia equivalente con valores normalizados de resistencias:

$$G = \left(1 + \frac{39k}{10k}\right) = 4,9 \tag{12.63}$$

Respecto a la imagen:

$$R22 = 39k \tag{12.64}$$
$$R24 = 0 \tag{12.65}$$
$$R27 = 10k \tag{12.66}$$
$$\tag{12.67}$$

12.3.5. Resistencia y LED de encendido

El único componente a dimensionar es la resistencia R1. Para ello se parte de los siguientes datos:

- Tensión de alimentación: $Vin = 24,5V$
- Tensión de polarización del LED del optoacoplador: $V_{LED} = 1,8V$
- Corriente a través del LED: $I_{LED} = 10mA$

De los que se deduce:

$$24,5V = V_R + V_{LED} \quad (12.68)$$
$$= I_{LED}R + V_{LED} \quad (12.69)$$
$$(12.70)$$

Despejando:

$$R = \frac{24,5V - V_{LED}}{I_{LED}} \quad (12.71)$$
$$= \frac{24,5V - 1,8V}{10mA} \quad (12.72)$$
$$= 2,27k\Omega \quad (12.73)$$
$$\mathbf{R = 2k2} \quad (12.74)$$
$$(12.75)$$

TÍTULO: **INSTALACIÓN DE UN BRAZO ROBOTIZADO PARA LA AUTOMATIZACIÓN DE LIMPIEZA CON LÁSER DE SUPERFICIES 3D**

PLANOS

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

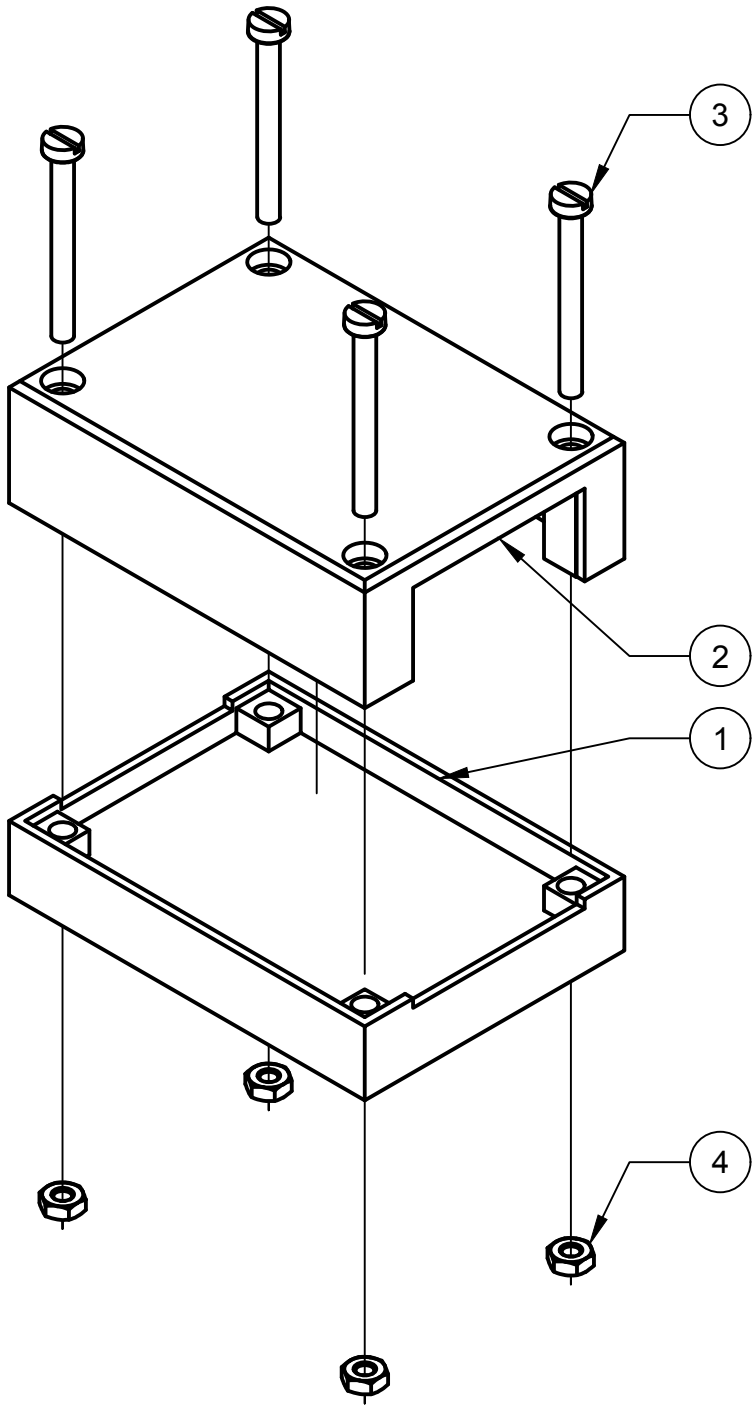
15405 - FERROL

FECHA: JUNIO DE 2019

AUTOR: EL ALUMNO

Fdo.: ADRIÁN CORA SIERRA

Índice de planos



DESPIECE		
ETIQUETA	**CANTIDAD**	**PIEZA**
1	1	PIEZA 1
2	1	PIEZA 2
3	4	TORNILLO M3X30
4	4	TUERCA HEXAGONAL M3

UNIVERSIDADE DA CORUÑA ESCUELA UNIVERSITARIA POLITÉCNICA

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TFG Nº: 770G01A162

TÍTULO DEL TFG: INSTALACIÓN DE UN BRAZO ROBOTIZADO PARA LA AUTOMATIZACIÓN DE LIMPIEZA CON LÁSER DE SUPERFICIES 3D

TÍTULO DEL PLANO:

EXPLOSIONADO CARCASA WIZ750SR-110

FECHA: JUNIO-2019

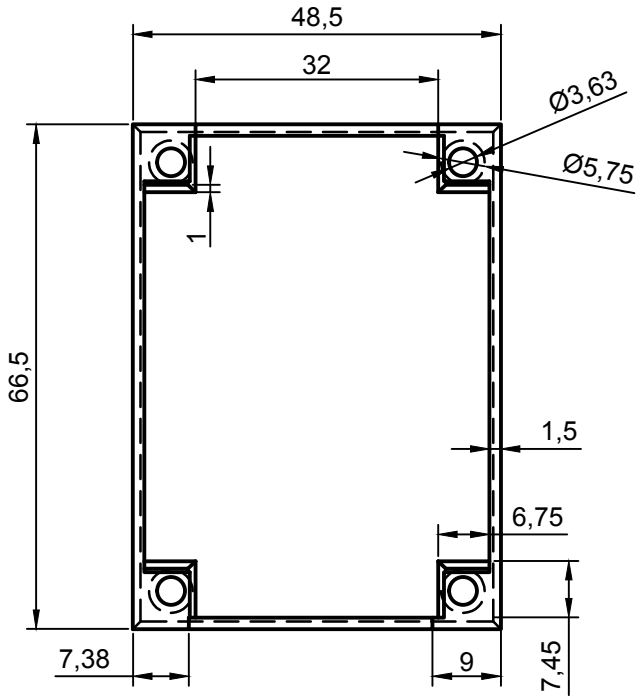
ESCALA: 1:1

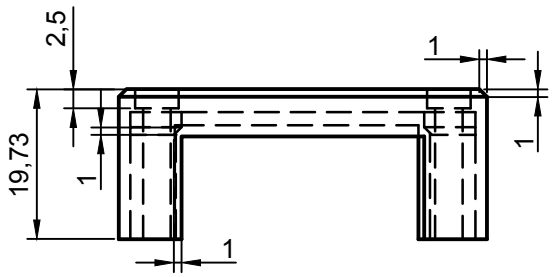
AUTOR:

ADRIÁN CORA SIERRA

FIRMA:

PLANO Nº: 01





UNIVERSIDADE DA CORUÑA ESCUELA UNIVERSITARIA POLITÉCNICA

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TFG Nº: 770G01A162

TÍTULO DEL TFG: INSTALACIÓN DE UN BRAZO ROBOTIZADO PARA LA AUTOMATIZACIÓN DE LIMPIEZA CON LÁSER DE SUPERFICIES 3D

TÍTULO DEL PLANO:

WIZ750SR-110 CARCASA 1

FECHA: JUNIO-2019

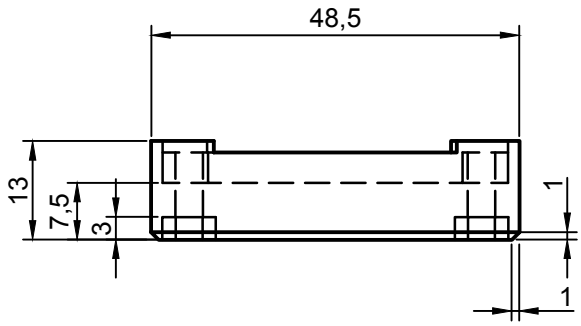
ESCALA: 1:1

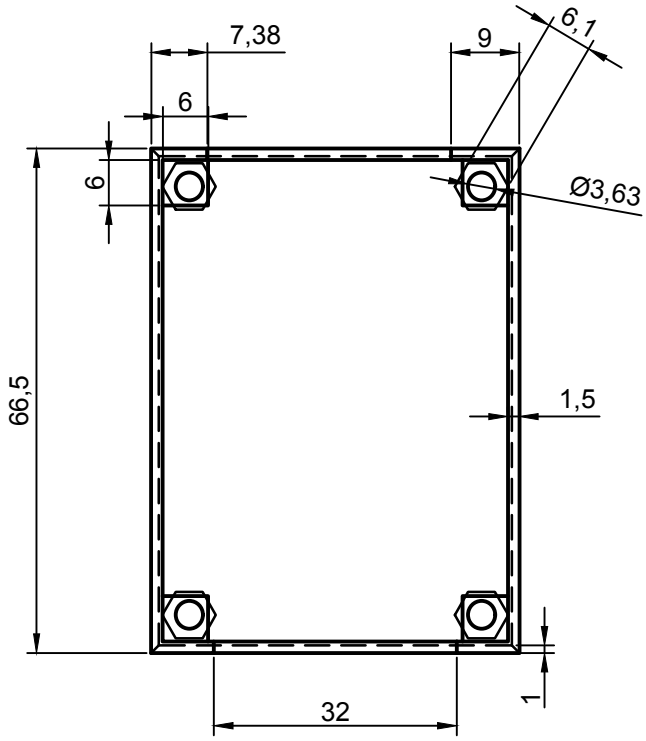
AUTOR:

ADRIÁN CORA SIERRA

FIRMA:

PLANO Nº: 02





UNIVERSIDADE DA CORUÑA ESCUELA UNIVERSITARIA POLITÉCNICA

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TFG Nº: 770G01A162

TÍTULO DEL TFG: INSTALACIÓN DE UN BRAZO ROBOTIZADO PARA LA AUTOMATIZACIÓN DE LIMPIEZA CON LÁSER DE SUPERFICIES 3D

TÍTULO DEL PLANO:

WIZ750SR-110 CARCASA 2

FECHA: JUNIO-2019

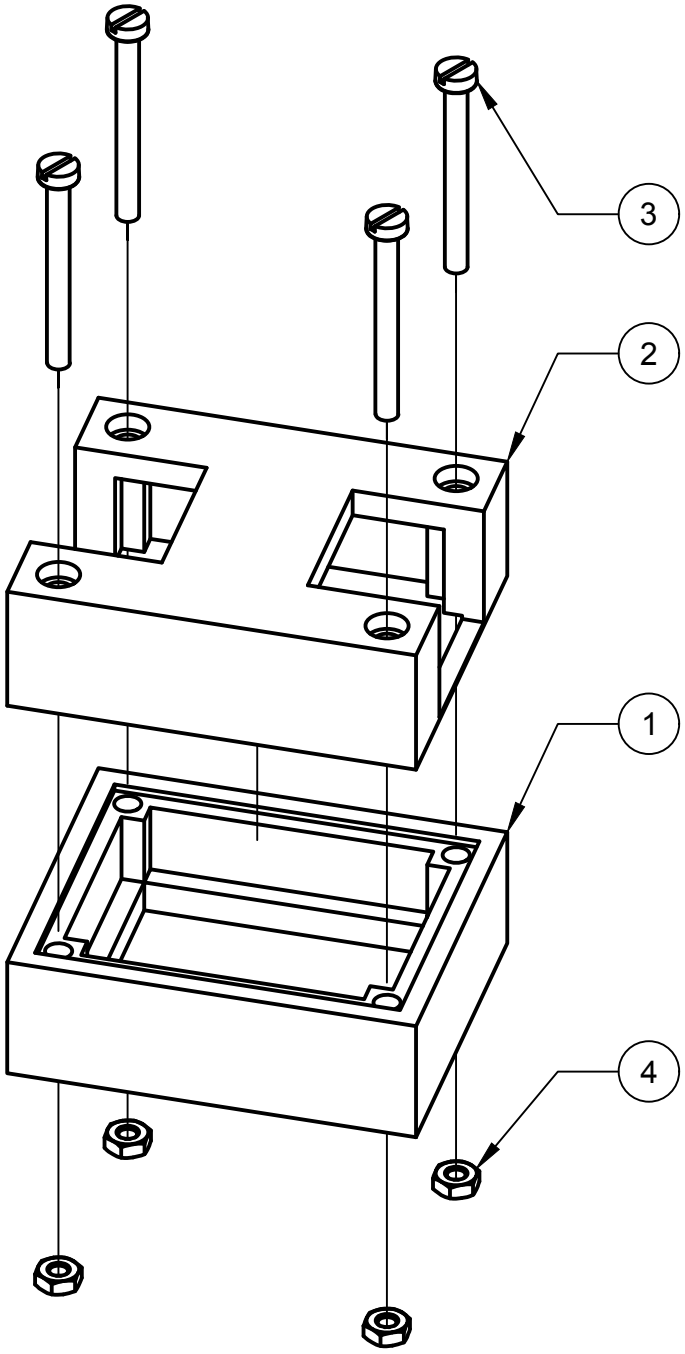
ESCALA: 1:1

AUTOR:

ADRIÁN CORA SIERRA

FIRMA:

PLANO Nº: 03



DESPIECE		
ETIQUETA	**CANTIDAD**	**PIEZA**
1	1	PIEZA 1
2	1	PIEZA 2
3	4	TORNILLO M3X30
4	4	TUERCA HEXAGONAL M3

UNIVERSIDADE DA CORUÑA ESCUELA UNIVERSITARIA POLITÉCNICA

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TFG Nº: 770G01A162

TÍTULO DEL TFG: INSTALACIÓN DE UN BRAZO ROBOTIZADO PARA LA AUTOMATIZACIÓN DE LIMPIEZA CON LÁSER DE SUPERFICIES 3D

TÍTULO DEL PLANO:

EXPLOSIONADO CARCASA OPTOACOPLADOR

FECHA: JUNIO-2019

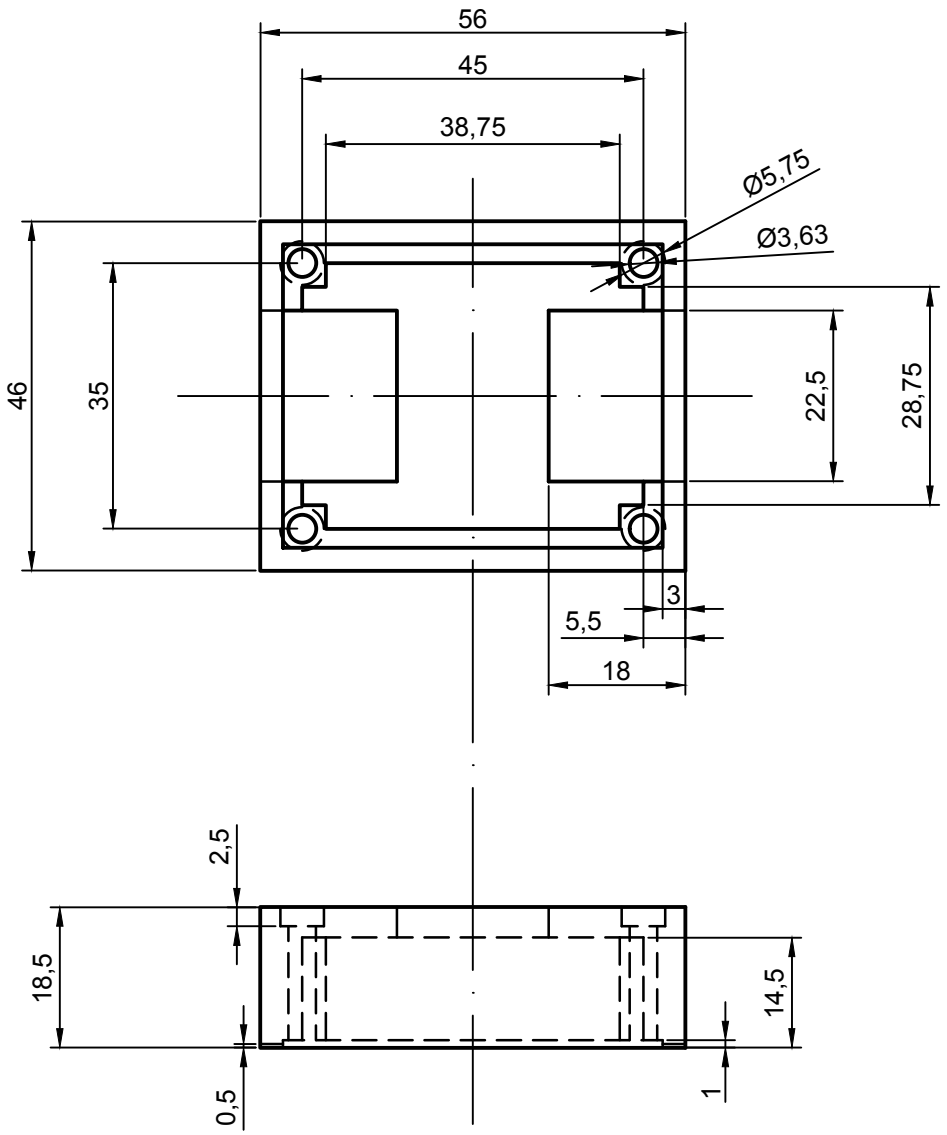
ESCALA: 1:1

AUTOR:

ADRIÁN CORA SIERRA

FIRMA:

PLANO Nº: 04



UNIVERSIDADE DA CORUÑA ESCUELA UNIVERSITARIA POLITÉCNICA

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TFG Nº: 770G01A162

TÍTULO DEL TFG: INSTALACIÓN DE UN BRAZO ROBOTIZADO PARA LA AUTOMATIZACIÓN DE LIMPIEZA CON LÁSER DE SUPERFICIES 3D

TÍTULO DEL PLANO:

OPTOACOPLADOR CARCASA 1

FECHA: JUNIO-2019

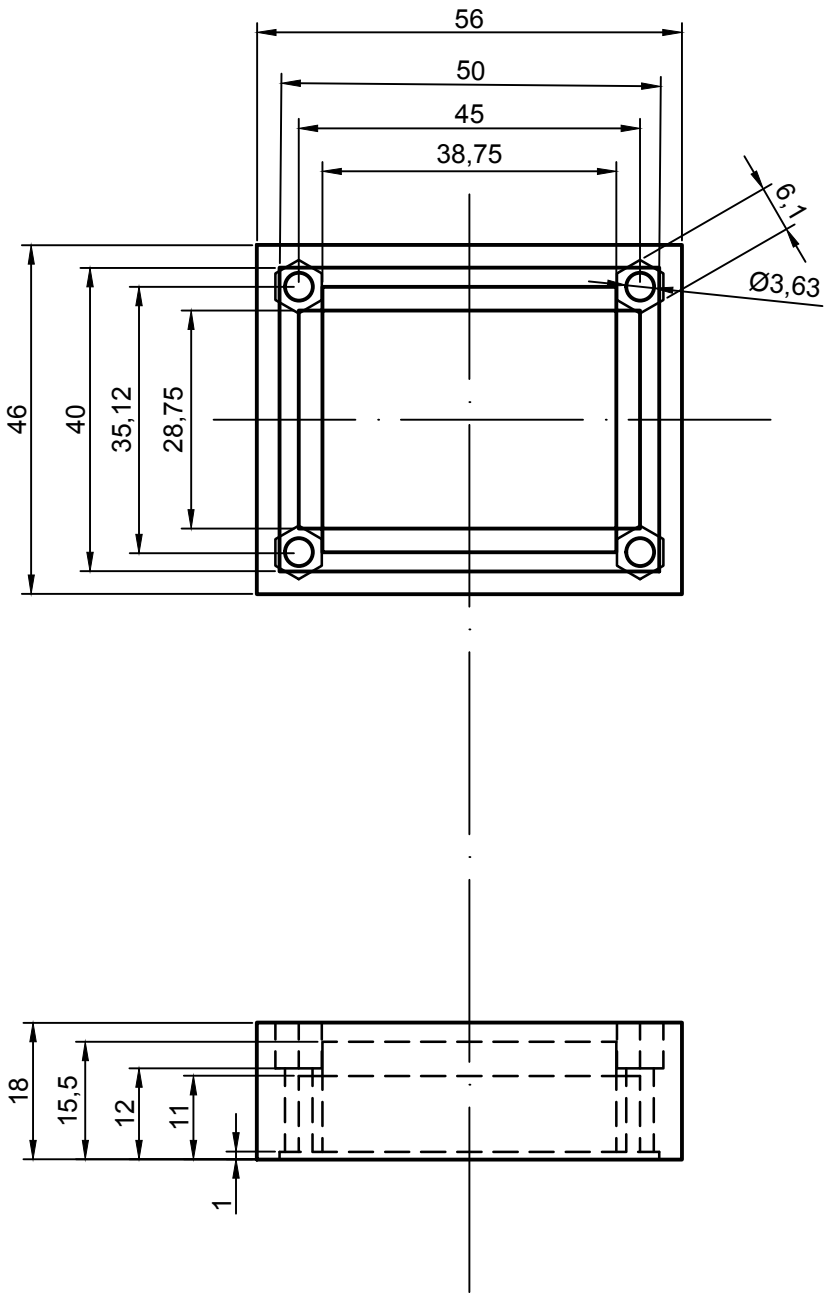
ESCALA: 1:1

AUTOR:

ADRIÁN CORA SIERRA

FIRMA:

PLANO Nº: 05



UNIVERSIDADE DA CORUÑA ESCUELA UNIVERSITARIA POLITÉCNICA

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TFG Nº: 770G01A162

TÍTULO DEL TFG: INSTALACIÓN DE UN BRAZO ROBOTIZADO PARA LA AUTOMATIZACIÓN DE LIMPIEZA CON LÁSER DE SUPERFICIES 3D

TÍTULO DEL PLANO:

OPTOACOPLADOR CARCASA 2

FECHA: JUNIO-2019

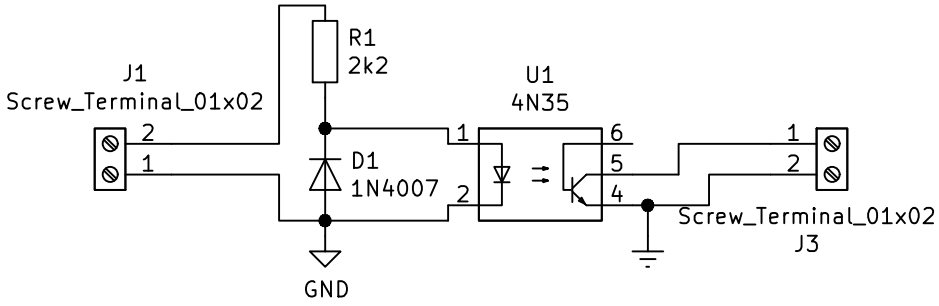
ESCALA: 1:1

AUTOR:

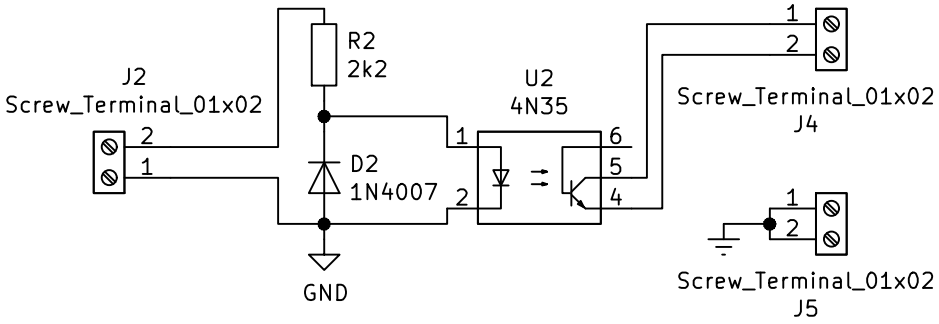
ADRIÁN CORA SIERRA

FIRMA:

PLANO Nº: 06



UNIVERSIDADE DA CORUÑA — ESCUELA UNIVERSITARIA POLITÉCNICA GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA	TFG Nº: 770G01A162
TÍTULO DEL TFG: INSTALACIÓN DE UN BRAZO ROBOTIZADO PARA LA AUTOMATIZACIÓN DE LIMPIEZA CON LÁSER DE SUPERFICIES 3D	
TÍTULO DEL PLANO: ESQUEMA OPTOACOPLADOR LÁSER AVIA	FECHA: JUNIO-2019
AUTOR: ADRIÁN CORA SIERRA FIRMA:	ESCALA: PLANO Nº: 07



UNIVERSIDADE DA CORUÑA ESCUELA UNIVERSITARIA POLITÉCNICA

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TFG Nº: 770G01A162

TÍTULO DEL TFG: INSTALACIÓN DE UN BRAZO ROBOTIZADO PARA LA AUTOMATIZACIÓN DE LIMPIEZA CON LÁSER DE SUPERFICIES 3D

TÍTULO DEL PLANO:

ESQUEMA OPTOACOPLADOR LÁSER SPIRIT

FECHA: JUNIO-2019

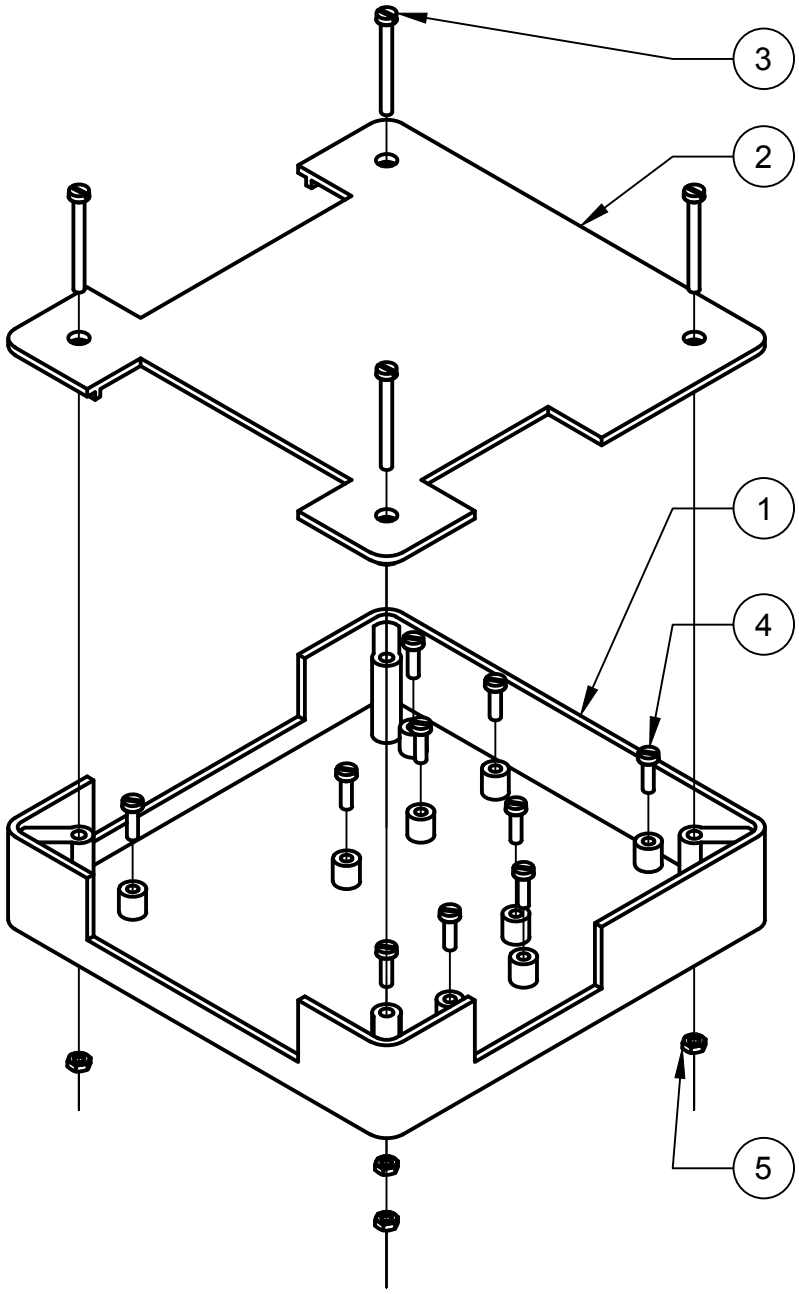
ESCALA:

AUTOR:

ADRIÁN CORA SIERRA

FIRMA:

PLANO Nº: 08



DESPIECE		
ETIQUETA	**CANTIDAD**	**PIEZA**
1	1	CARCASA 1
2	1	CARCASA 2
3	4	TORNILLO M3X30
4	10	TORNILLO M3X10
5	4	TUERCA HEXAGONAL M3

UNIVERSIDADE DA CORUÑA ESCUELA UNIVERSITARIA POLITÉCNICA

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TFG Nº: 770G01A162

TÍTULO DEL TFG: INSTALACIÓN DE UN BRAZO ROBOTIZADO PARA LA AUTOMATIZACIÓN DE LIMPIEZA CON LÁSER DE SUPERFICIES 3D

TÍTULO DEL PLANO:

EXPLOSIONADO CARCASA DAQ ARDUINO

FECHA: JUNIO-2019

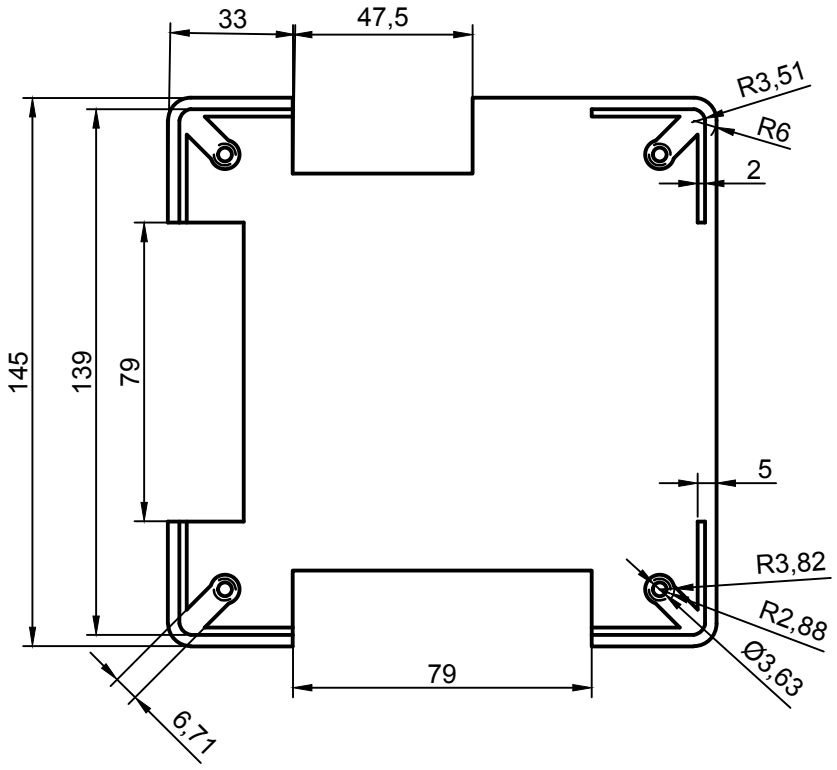
ESCALA: 1:2

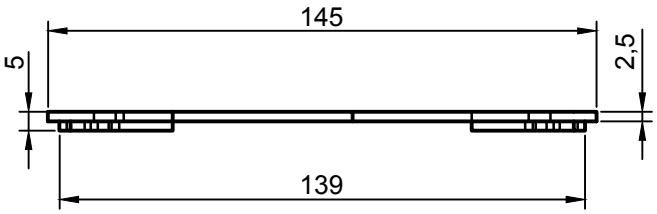
AUTOR:

ADRIÁN CORA SIERRA

FIRMA:

PLANO Nº: 09





UNIVERSIDADE DA CORUÑA ESCUELA UNIVERSITARIA POLITÉCNICA

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TFG Nº: 770G01A162

TÍTULO DEL TFG: INSTALACIÓN DE UN BRAZO ROBOTIZADO PARA LA AUTOMATIZACIÓN DE LIMPIEZA CON LÁSER DE SUPERFICIES 3D

TÍTULO DEL PLANO:

DAQ ARDUINO CARCASA 2

FECHA: JUNIO-2019

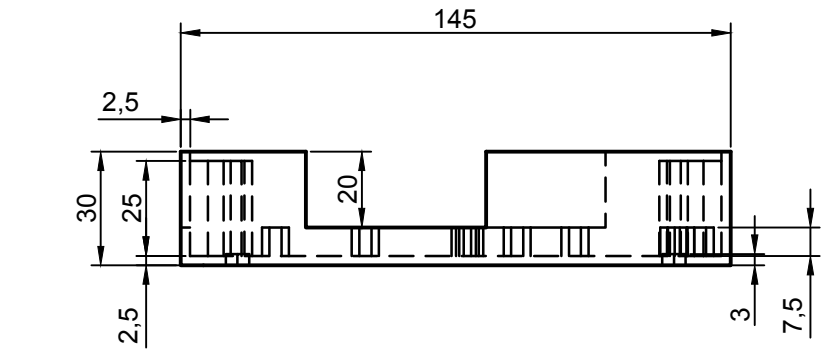
ESCALA: 1:2

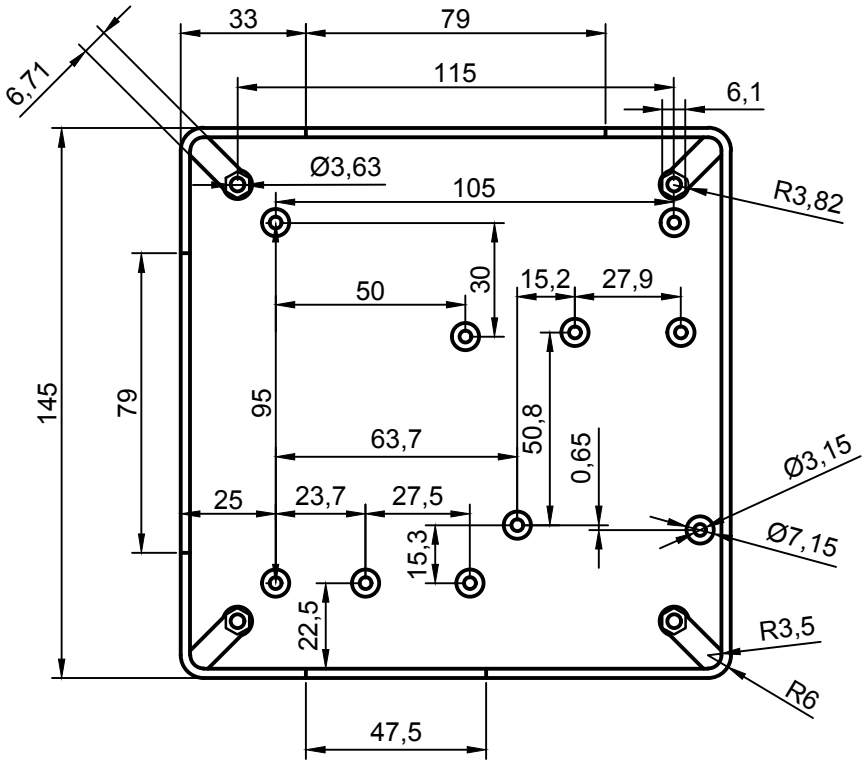
AUTOR:

ADRIÁN CORA SIERRA

FIRMA:

PLANO Nº: 10





UNIVERSIDADE DA CORUÑA ESCUELA UNIVERSITARIA POLITÉCNICA

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TFG Nº: 770G01A162

TÍTULO DEL TFG: INSTALACIÓN DE UN BRAZO ROBOTIZADO PARA LA AUTOMATIZACIÓN DE LIMPIEZA CON LÁSER DE SUPERFICIES 3D

TÍTULO DEL PLANO:

DAQ ARDUINO CARCASA 1

FECHA: JUNIO-2019

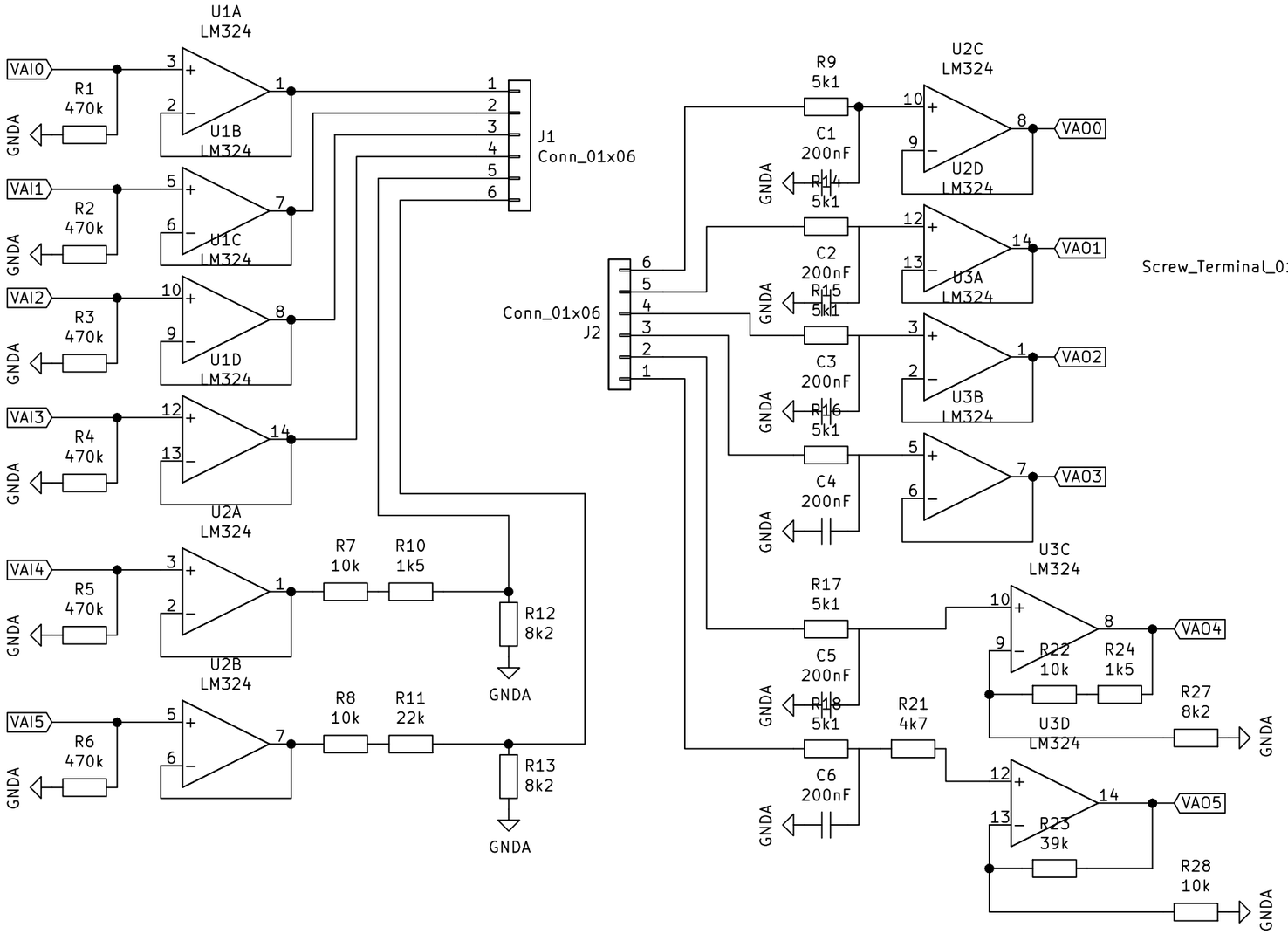
ESCALA: 1:2

AUTOR:

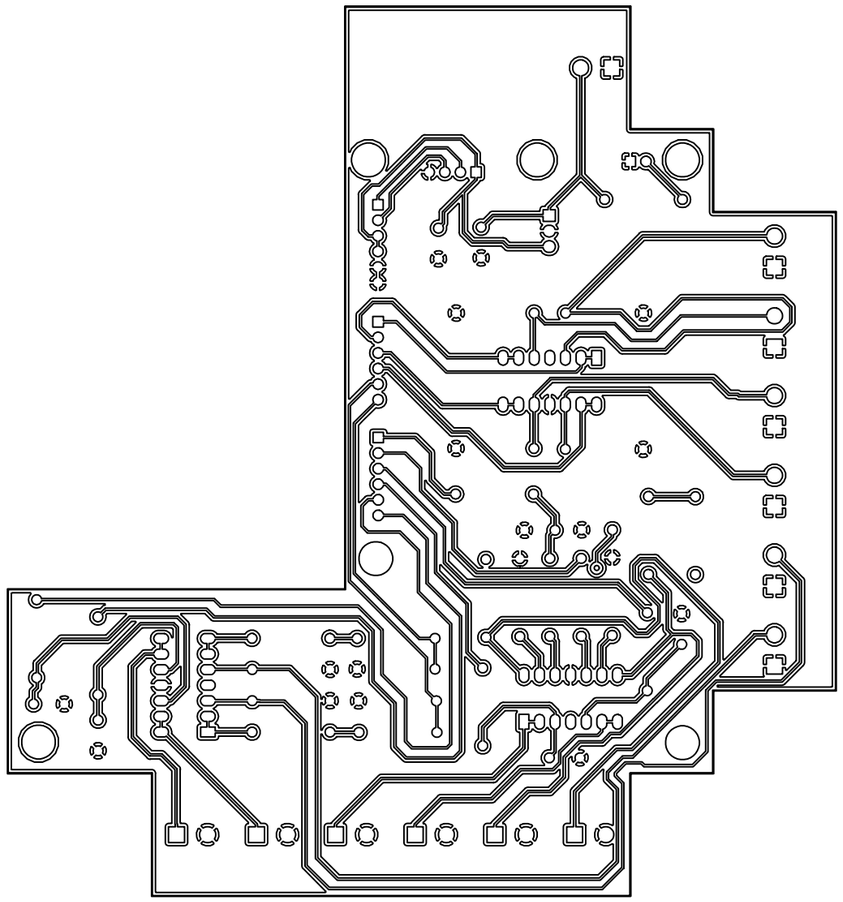
ADRIÁN CORA SIERRA

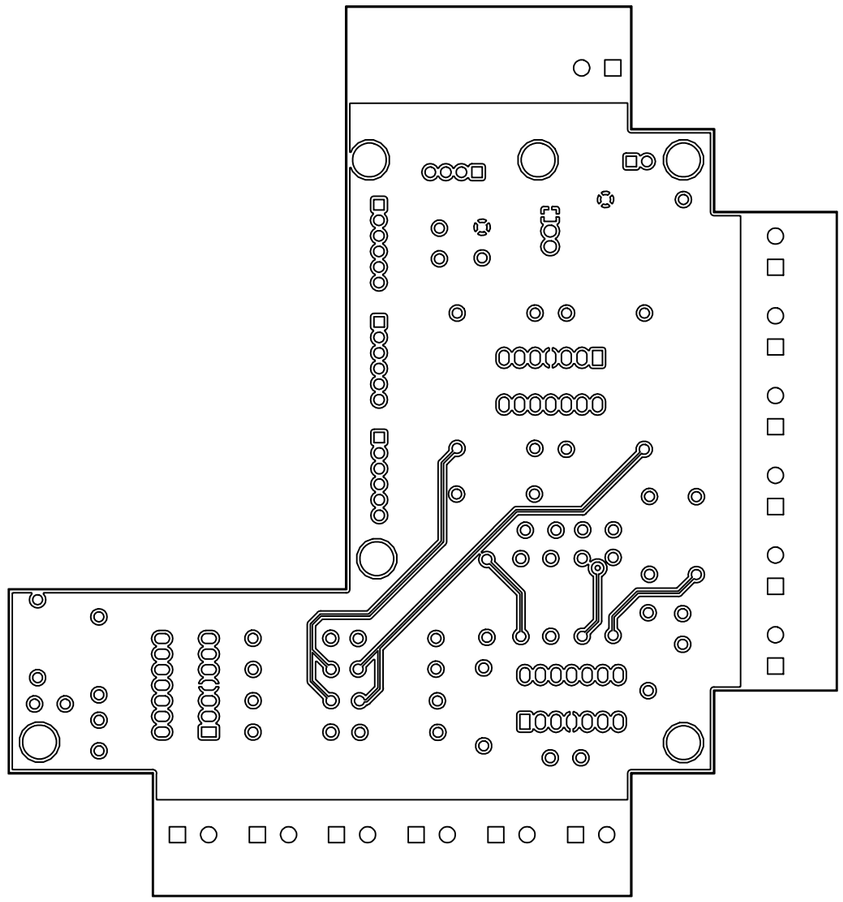
FIRMA:

PLANO Nº: 11



UNIVERSIDADE DA CORUÑA — ESCUELA UNIVERSITARIA POLITÉCNICA GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA	TFG Nº: 770G01A162
TÍTULO DEL TFG: INSTALACIÓN DE UN BRAZO ROBOTIZADO PARA LA AUTOMATIZACIÓN DE LIMPIEZA CON LÁSER DE SUPERFICIES 3D	
TÍTULO DEL PLANO: ESQUEMA DAQ ARDUINO	FECHA: JUNIO-2019
AUTOR: ADRIÁN CORA SIERRA / FIRMA:	ESCALA:
	PLANO Nº: 12





UNIVERSIDADE DA CORUÑA ESCUELA UNIVERSITARIA POLITÉCNICA

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TFG Nº: 770G01A162

TÍTULO DEL TFG: INSTALACIÓN DE UN BRAZO ROBOTIZADO PARA LA AUTOMATIZACIÓN DE LIMPIEZA CON LÁSER DE SUPERFICIES 3D

TÍTULO DEL PLANO:

PCB DAQ ARDUINO RELLENOS COBRE [F,B]

FECHA: JUNIO-2019

ESCALA: 1:1

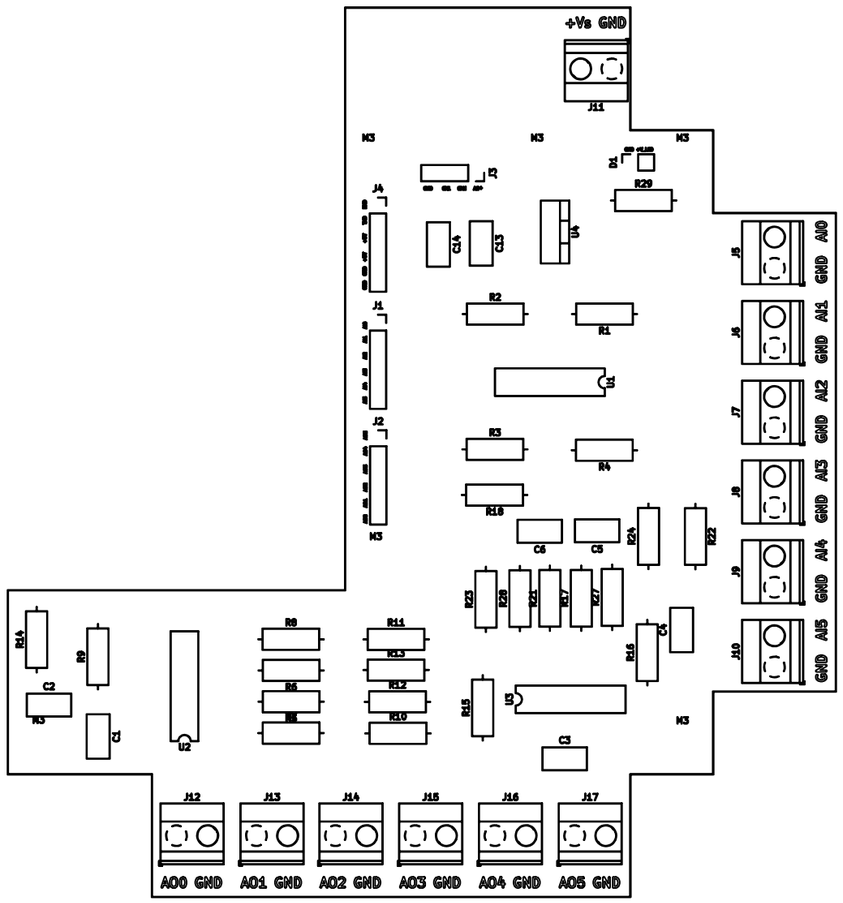
AUTOR:

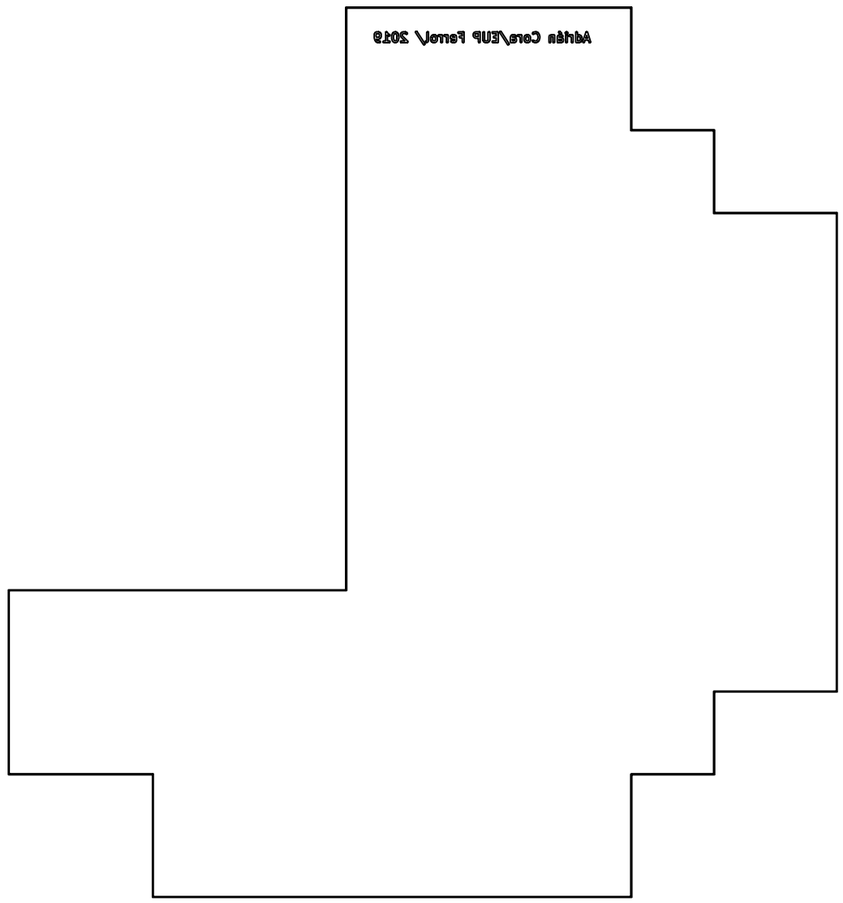
ADRIÁN CORA SIERRA

FIRMA:

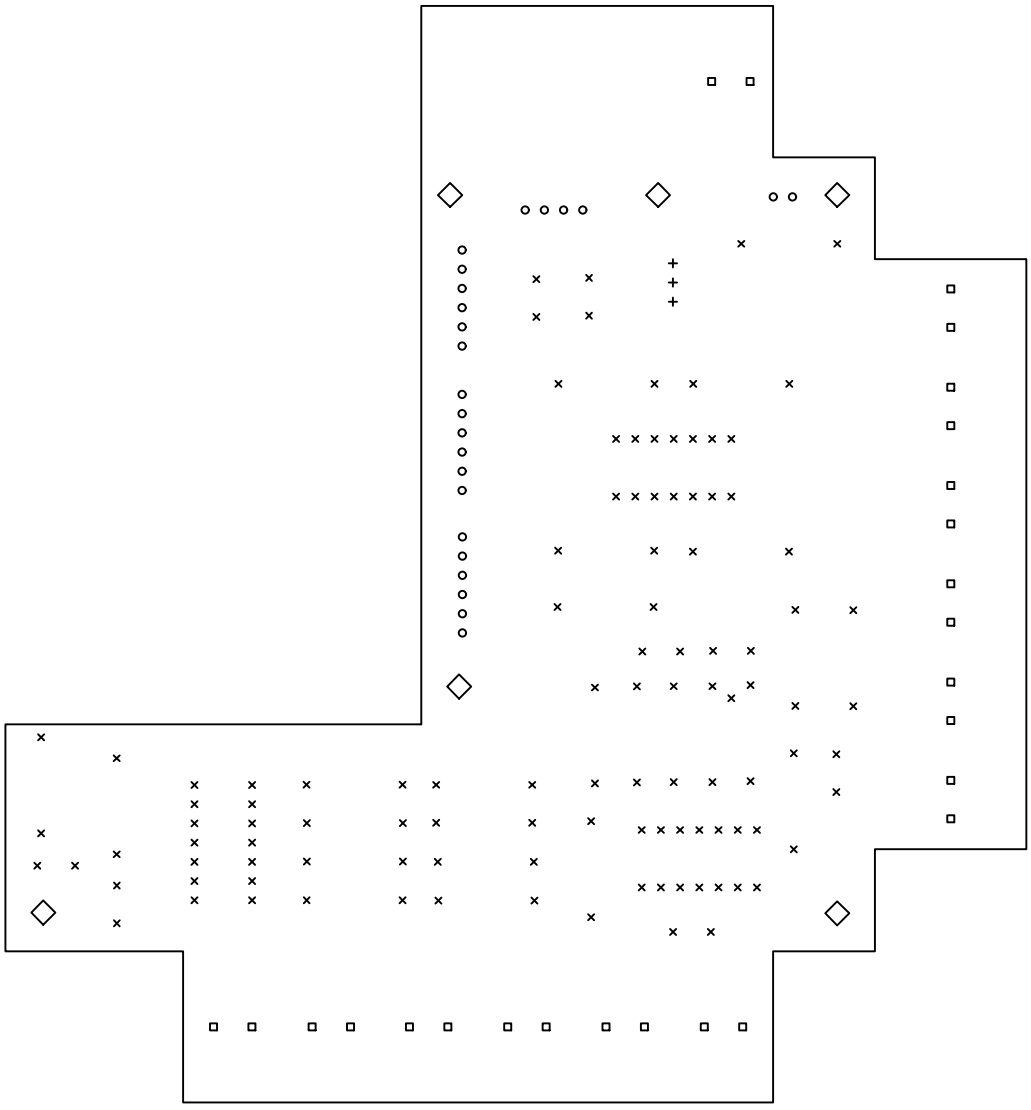
PLANO Nº: 13

A3





UNIVERSIDADE DA CORUÑA — ESCUELA UNIVERSITARIA POLITÉCNICA GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA	TFG Nº: 770G01A162
TÍTULO DEL TFG: INSTALACIÓN DE UN BRAZO ROBOTIZADO PARA LA AUTOMATIZACIÓN DE LIMPIEZA CON LÁSER DE SUPERFICIES 3D	
TÍTULO DEL PLANO: PCB DAQ ARDUINO SERIGRAFÍAS [F,B]	FECHA: JUNIO-2019 ESCALA: 1:1
AUTOR: ADRIÁN CORA SIERRA FIRMA:	PLANO Nº: 14



Drill Map:

× 0.80mm / 0.031" (109 holes)
∘ 1.00mm / 0.039" (24 holes)
+ 1.10mm / 0.043" (3 holes)
▫ 1.30mm / 0.051" (26 holes)
◇ 3.20mm / 0.126" (6 holes)

UNIVERSIDADE DA CORUÑA — ESCUELA UNIVERSITARIA POLITÉCNICA

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TFG Nº: 770G01A162

TÍTULO DEL TFG: INSTALACIÓN DE UN BRAZO ROBOTIZADO PARA LA AUTOMATIZACIÓN DE LIMPIEZA CON LÁSER DE SUPERFICIES 3D

TÍTULO DEL PLANO:

PCB DAQ ARDUINO TALADROS [F]

FECHA: JUNIO-2019

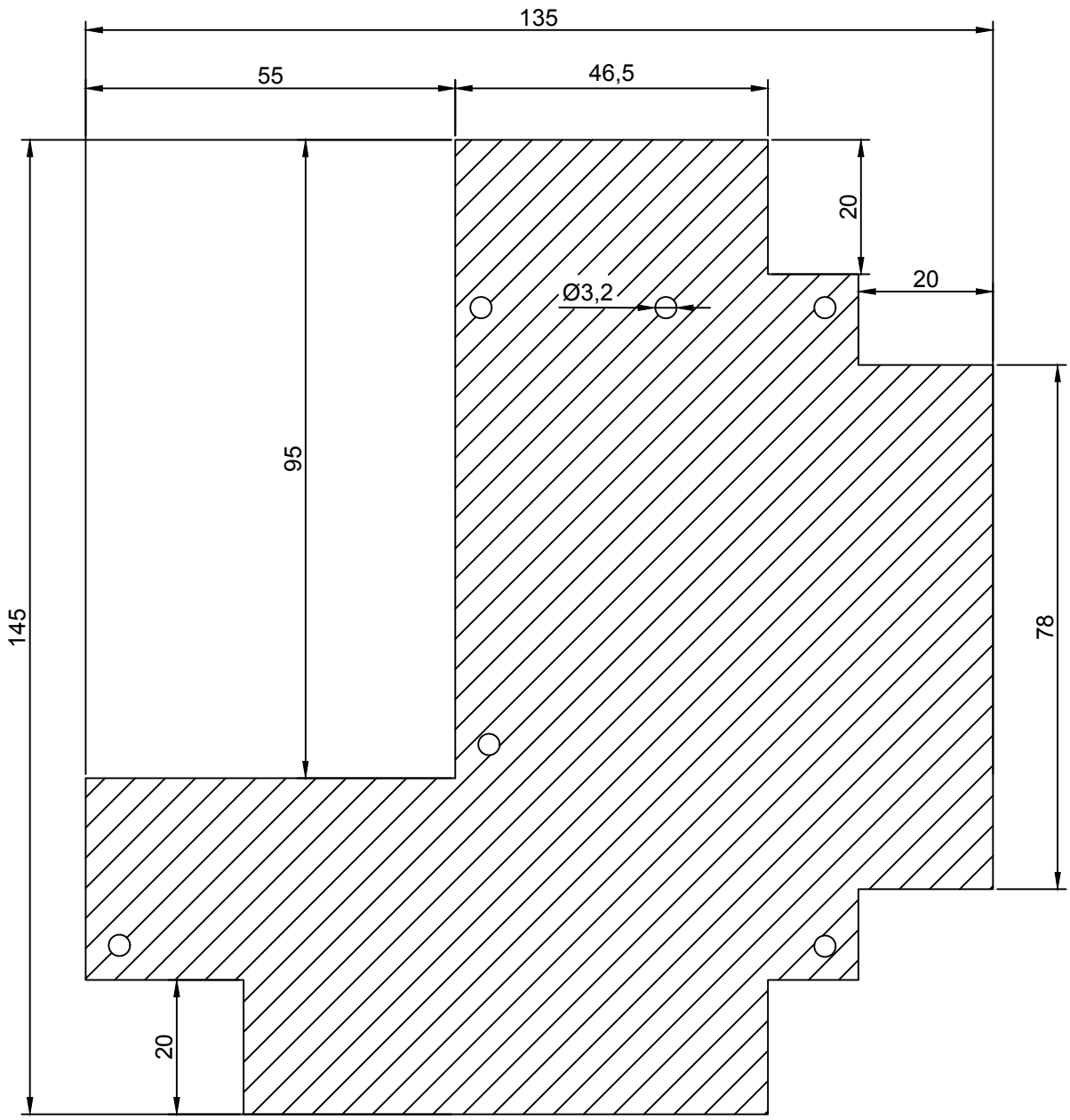
ESCALA: 1:1

AUTOR:

ADRIÁN CORA SIERRA

FIRMA:

PLANO Nº: 15



UNIVERSIDADE DA CORUÑA ESCUELA UNIVERSITARIA POLITÉCNICA

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TFG Nº: 770G01A162

TÍTULO DEL TFG: INSTALACIÓN DE UN BRAZO ROBOTIZADO PARA LA AUTOMATIZACIÓN DE LIMPIEZA CON LÁSER DE SUPERFICIES 3D

TÍTULO DEL PLANO:

PCB DAQ ARDUINO DIMENSIONES

FECHA: JUNIO-2019

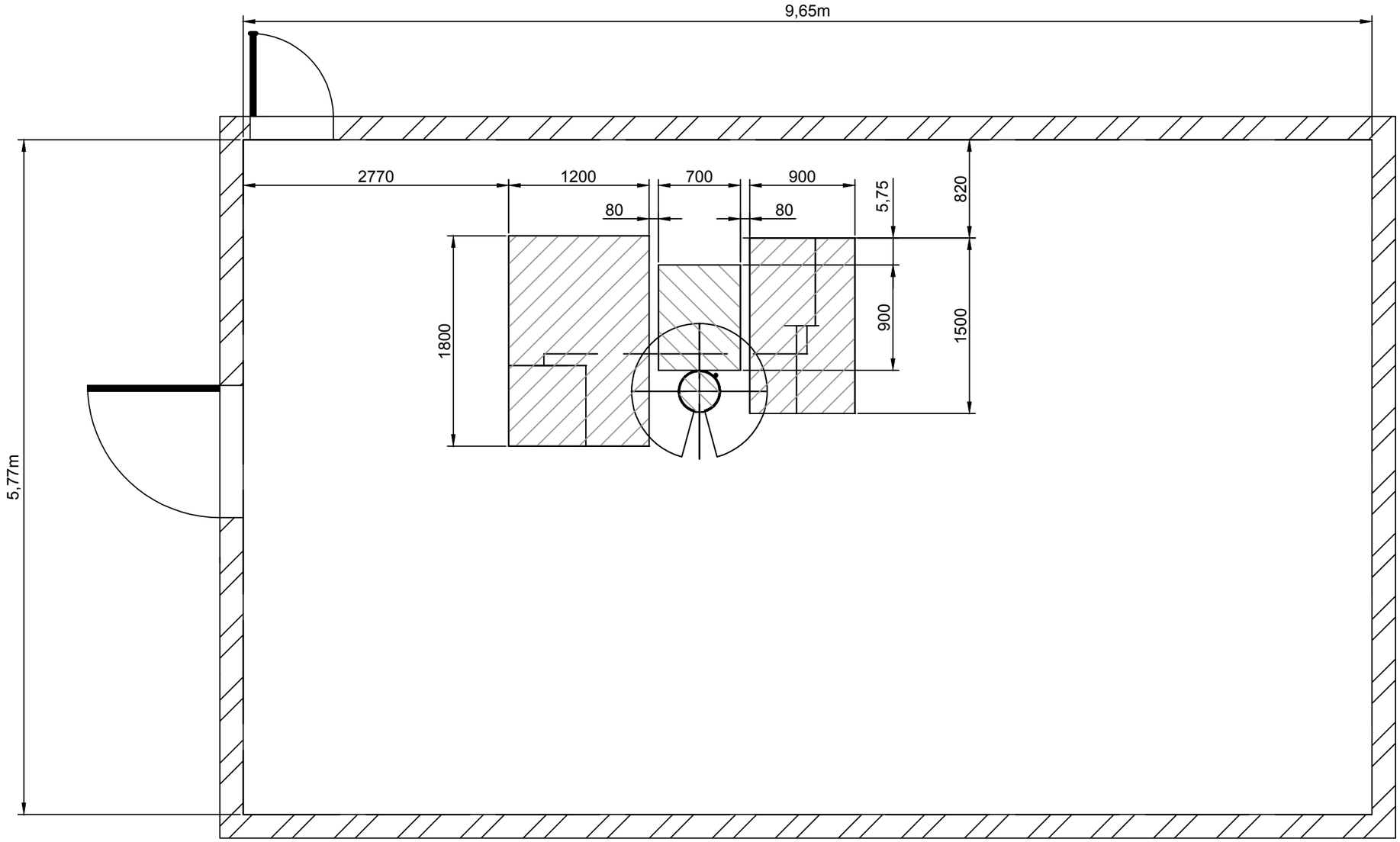
ESCALA: 1:1

AUTOR:

ADRIÁN CORA SIERRA

FIRMA:

PLANO Nº: 16



UNIVERSIDADE DA CORUÑA — ESCUELA UNIVERSITARIA POLITÉCNICA GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA	TFG Nº: 770G01A162
TÍTULO DEL TFG: INSTALACIÓN DE UN BRAZO ROBOTIZADO PARA LA AUTOMATIZACIÓN DE LIMPIEZA CON LÁSER DE SUPERFICIES 3D	
TÍTULO DEL PLANO: DISTRIBUCIÓN DE LOS EQUIPOS	FECHA: JUNIO-2019 ESCALA: 1:40
AUTOR: ADRIÁN CORA SIERRA FIRMA:	PLANO Nº: 17

A3

TÍTULO: **INSTALACIÓN DE UN BRAZO ROBOTIZADO PARA LA AUTOMATIZACIÓN DE LIMPIEZA CON LÁSER DE SUPERFICIES 3D**

PLIEGO DE CONDICIONES

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: JUNIO DE 2019

AUTOR: EL ALUMNO

Fdo.: ADRIÁN CORA SIERRA

Índice del documento PLIEGO DE CONDICIONES

13 PLIEGO DE CONDICIONES

13.1. Especificaciones de Software

- La versión de Python sobre la que corre el software implementado en este lenguaje es Python 3.7.1 o superior.
- El sistema operativo sobre el que se ejecutan los códigos en Python es Microsoft Windows 7, 8 o 10. De no ser así pueden aparecer problemas con la librería *pyserial*.
- Los códigos en RAPID son compatibles con RobotWare 6.08 o superior.

13.2. Especificaciones de Hardware

- Para la impresión 3D de las piezas correspondientes de este trabajo se recomienda emplear el material PLA (Ácido Poliláctico).
- Toda la tornillería necesaria para el ensamblaje de las piezas impresas en 3D es de los tipos:
  - Tornillo M3x30 alomado con cabeza plana.
  - Tornillo M3x10 alomado con cabeza plana.
  - Tuerca hexagonal M3.
- Se recomienda realizar el cableado de señales analógicas y digitales con cable coaxial.
- La alimentación de la DAQ Arduino se puede realizar con la fuente da alimentación de 24,5V DC de la controladora del brazo robot (IRC5 Compact).
- Es necesario cablear la alimentación de la tarjeta de salidas digitales (DSQC 652) de de la controladora del brazo robot (IRC5 Compact) de manera externa.
- Se recomienda la instalación de un disipador con pasta térmica en el regulador de tensión lineal LM7805 de la DAQ Arduino.
- En la reproducción de la DAQ Arduino es necesario emplear la versión de la placa Arduino, Arduino UNO R3 (ATMega 328p). En caso de no ser así, es probable que el código no sea compatible.
- En la fabricación de la PCB se recomienda:
  - El uso de zócalos para la conexión de los circuitos integrados LM324. Se pueden dañar al soldarlos de forma directa en la placa.
  - Se recomienda que el grosor de las pistas de cobre sea de 1oz y el de la placa PCB de 1,6mm.

13.3. Condiciones de almacenamiento

Las condiciones para el almacenamiento para el conjunto son las indicadas por el fabricante de cada uno de los componentes de los que se disponga documentación. En caso de ambigüedad las condiciones han de ser las más restrictivas.

De manera más generalista se recomienda el almacenamiento en lugares cuya temperatura no supere los 30ºC, sin polvo, sin luz sólar directa y con una baja húmedad relativa.

TÍTULO: **INSTALACIÓN DE UN BRAZO ROBOTIZADO PARA LA AUTOMATIZACIÓN DE LIMPIEZA CON LÁSER DE SUPERFICIES 3D**

MEDICIONES

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: JUNIO DE 2019

AUTOR: EL ALUMNO

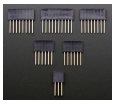
Fdo.: ADRIÁN CORA SIERRA

Índice del documento MEDICIONES

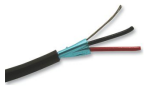
14 Materiales

Pasarelas WIZ750SR-110				
ID	**Imagen**	**Descripción**	**Referencia**	**Ud**
WIZ750SR-110		Placa WIZ750SR-100	Fabricante: WIZ110SR	2
FA-WIZ		Fuente de alimentación 5VDC Jack	Farnell: 2451858	2
ET-WIZ		Cable ethernet RJ45 1m	Farnell: 2056746	2
RS-WIZ		Cable RS-232 cruzado	Farnell: 2444240	2

Optoacopladores Láseres AVIA y Spirit				
ID	**Imagen**	**Descripción**	**Referencia**	**Ud**
J1,J2,J3,J4,J5		Bloque Terminal Cable a Placa, 5.08 mm, 2 Vías	Farnell: 3041165	8
R1,R2		Resistencia de Agujero Pasante, 2.2 k, 250 mW	Farnell: 9339302	10
D1,D2		Diodo 1N4007 1 kV, 1 A, 480 mV,30 A	Farnell: 1651091	5
U1,U2		Optoacoplador 4N35, DIP 6 Pines	Farnell: 1328375	5
PCB_Prot		Placa de Prototipado, Cobre Estañado	Farnell: 1201473	1

DAQ Arduino (Parte 1)				
ID	**Imagen**	**Descripción**	**Referencia**	**Ud**
J16,J5,J6,J7,J8, J9,J10,J11,J12, J13,J14,J15,J17		Bloque Terminal Cable a Placa, 5.08 mm, 2 Vías	Farnell: 3041165	15
C1,C2,C3, C4,C5,C6		Condensador de Película DC, 0.22 µF, 100 V, PET	Farnell: 9753001	8
C13		Condensador de Película DC, 0.33 µF, 100 V, PET	Farnell: 2525327	5
C14		Condensador de Película DC, 0.1 µF, 100 V, PET	Farnell: 2525325	5
D1		LED, Rojo, Agujero Pasante, 20 mA, 1.85 V	Farnell: 2335728	5
J1,J2,J3,J4		Accessory Type:Shield Stacking Headers	Farnell: 2816284	1
R1,R2,R3, R4,R5,R6		R de Agujero Pasante, 470 k, 250 mW	Farnell: 9339566	6
R7,R8, R22,R28		R de Agujero Pasante, 10 k, 250 mW	Farnell: 9339060	10
R9,R14,R15, R16,R17,R18		R de Agujero Pasante, 5.1 k, 250 mW	Farnell: 9342010	10
R10,R24		R de Agujero Pasante, 1.5 k, 250 mW	Farnell: 9339183	10
R11		R de Agujero Pasante, 22 k, 250 mW	Farnell: 9341544	10
R12,R13,R27		R de Agujero Pasante, 8.2 k, 250 mW	Farnell: 9342273	10
R21		R de Agujero Pasante, 4.7 k, 250 mW	Farnell: 9339540	10

DAQ Arduino (Parte 2)				
ID	**Imagen**	**Descripción**	**Referencia**	**Ud**
R23		R de Agujero Pasante, 39 k, 250 mW	Farnell: 9341862	10
R29		R de Agujero Pasante, 1.2 k, 250 mW	Farnell: 9339124	10
U1,U2,U3		LM324N 4 AO DIP-14	Farnell: 1417640	5
SD14		Zócalo CI DIP-14 Socket DIP	Farnell: 2672296	10
U4		LM7805 Regulador Lineal de Tensión	Farnell: 2296047	1
PCB		Placa PCB	Fabricante: JLC PCB	5
A_UNO_R3		Arduino Uno, MCU ATmega328P	Farnell: 2075382	1
RS232_TTL		Conversor RS232 a TTL	e-ika: SKU: 1330	2

Cableado				
ID	**Imagen**	**Descripción**	**Referencia**	**Ud**
D-Sub		Conector D-Sub Combinado, DC-21WA4	Farnell: 2433176	1
BNC		Conector de RF/Coaxial, Coaxial BNC, Macho Recto, Soldable, 50 ohm	Farnell: 1205943	3
TRMNL		Virola para Cable, Terminación DIN, Cable Simple, 20 AWG, 0.5 mm2, 8 mm, Blanco	Farnell: 1772646	100
CCXL		Cable Coaxial, 0.14 mm2, 50 ohm, 30.5 m	Farnell: 1302764	1
CMPT		Cable Multipar, Blindado, 1 Par, 22 AWG	Farnell: 2948817	1

Impresión 3D y Tornillería				
ID	**Imagen**	**Descripción**	**Referencia**	**Ud**
TT1		Tornillo, M3, 10 mm, Cabeza Cilíndrica y Plana Ranurada	Farnell: 1419787	1
TT2		Tornillo, M3, 30 mm, A2, Ranurada con Cabeza	Farnell: 1419348	1
TT3		Tuerca, M3, Hexagonal	Farnell: 1419447	1
PLA1		Filamento de Impresora 3D, 1.75mm, 1Kg, PLA, Blanco	Farnell: 2828655	1

15 Mano de obra

Mano de Obra (Parte 1)		
Tarea	**Personal**	**Horas**
Análisis de soluciones y documentación		
Documentación	Graduado en Ingeniería Electrónica Industrial y Automática	57
Análisis de soluciones	Graduado en Ingeniería Electrónica Industrial y Automática	29
Ejecución		
Distribución Equipos	Graduado en Ingeniería Electrónica Industrial y Automática	4
Programación Python	Graduado en Ingeniería Electrónica Industrial y Automática	25
Programación RAPID	Graduado en Ingeniería Electrónica Industrial y Automática	6
Programación Arduino	Graduado en Ingeniería Electrónica Industrial y Automática	8
Configuración WIZ750SR-110	Graduado en Ingeniería Electrónica Industrial y Automática	15
DAQ Arduino diseño y calculos	Graduado en Ingeniería Electrónica Industrial y Automática	20
Optoacopladores diseño y calculos	Graduado en Ingeniería Electrónica Industrial y Automática	20
CAD carcasas	Graduado en Ingeniería Electrónica Industrial y Automática	15

Mano de Obra (Parte 2)		
Tarea	**Personal**	**Horas**
Ejecución		
CAD PCB	Graduado en Ingeniería Electrónica Industrial y Automática	25
Soldadura y cableado	Graduado en Ingeniería Electrónica Industrial y Automática	5
Impresión 3D	Graduado en Ingeniería Electrónica Industrial y Automática	15
Pruebas		
Optoacopladores	Graduado en Ingeniería Electrónica Industrial y Automática	4
DAQ Arduino	Graduado en Ingeniería Electrónica Industrial y Automática	16
WIZ750SR-110	Graduado en Ingeniería Electrónica Industrial y Automática	8
Pruebas finales	Graduado en Ingeniería Electrónica Industrial y Automática	5
Redacción		
Redacción	Graduado en Ingeniería Electrónica Industrial y Automática	60

TÍTULO: **INSTALACIÓN DE UN BRAZO ROBOTIZADO PARA LA AUTOMATIZACIÓN DE LIMPIEZA CON LÁSER DE SUPERFICIES 3D**

PRESUPUESTO

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: JUNIO DE 2019

AUTOR: EL ALUMNO

Fdo.: ADRIÁN CORA SIERRA

Índice del documento PRESUPUESTO

16 Materiales

Pasarelas WIZ750SR-110				
ID	**Descripción**	**Ud**	**Precio Ud (€)**	**Precio total (€)**
WIZ750SR-110	Placa WIZ750SR-100	2	27,37	54,74
FA-WIZ	Fuente de alimentación 5VDC Jack	2	6,08	12,16
ET-WIZ	Cable ethernet RJ45 1m	2	1,27	2,54
RS-WIZ	Cable RS-232 cruzado	2	4,8	9,6
			TOTAL	79,04

Optoacopladores Láseres AVIA y Spirit				
ID	**Descripción**	**Ud**	**Precio Ud (€)**	**Precio total (€)**
J1,J2,J3,J4,J5	Bloque Terminal Cable a Placa, 5.08 mm, 2 Vías	8	1,18	9,44
R1,R2	Resistencia de Agujero Pasante, 2.2 k, 250 mW	10	0,0245	0,245
D1,D2	Diodo 1N4007 1 kV, 1 A, 480 mV,30 A	5	0,15	0,75
U1,U2	Optoacoplador 4N35, DIP 6 Pines	5	0,429	2,145
PCB_Prot	Placa de Prototipado, Cobre Estañado	1	6,01	6,01
			TOTAL	18,59

DAQ Arduino (Parte 1)				
ID	**Descripción**	**Ud**	**Precio Ud (€)**	**Precio total (€)**
J16,J5,J6,J7,J8, J9,J10,J11,J12, J13,J14,J15,J17	Bloque Terminal Cable a Placa, 5.08 mm, 2 Vías	15	1,18	17,7
C1,C2,C3, C4,C5,C6	Condensador de Película DC, 0.22 µF, 100 V, PET	8	0,488	3,904
C13	Condensador de Película DC, 0.33 µF, 100 V, PET	5	0,469	2,345
C14	Condensador de Película DC, 0.1 µF, 100 V, PET	5	0,469	2,345
D1	LED, Rojo, Agujero Pasante, 20 mA, 1.85 V	5	0,173	2,345
J1,J2,J3,J4	Accessory Type:Shield Stacking Headers	1	2,01	2,345
R1,R2,R3, R4,R5,R6	R de Agujero Pasante, 470 k, 250 mW	6	0,025	2,345
R7,R8, R22,R28	R de Agujero Pasante, 10 k, 250 mW	10	0,00247	2,345
R9,R14,R15, R16,R17,R18	R de Agujero Pasante, 5.1 k, 250 mW	10	0,0362	2,345
R10,R24	R de Agujero Pasante, 1.5 k, 250 mW	10	0,0245	2,345
R11	R de Agujero Pasante, 22 k, 250 mW	10	0,0344	2,345
R12,R13,R27	R de Agujero Pasante, 8.2 k, 250 mW	10	0,0352	2,345
R21	R de Agujero Pasante, 4.7 k, 250 mW	10	0,024	2,345

DAQ Arduino (Parte 2)				
ID	**Descripción**	**Ud**	**Precio Ud (€)**	**Precio total (€)**
R23	R de Agujero Pasante, 39 k, 250 mW	10	0,0362	0,362
R29	R de Agujero Pasante, 1.2 k, 250 mW	10	0,0245	0,245
U1,U2,U3	LM324N 4 AO DIP-14	5	0,472	2,36
SD14	Zócalo CI DIP-14 Socket DIP	10	0,3	3
U4	LM7805 Regulador Lineal de Tensión	1	0,782	0,782
PCB	Placa PCB	5	5,84	29,2
A_UNO_R3	Arduino Uno, MCU ATmega328P	1	18,69	18,69
RS232_TTL	Conversor RS232 a TTL	2	1,41	2,82
			TOTAL	88,35

Cableado				
ID	**Descripción**	**Ud**	**Precio Ud (€)**	**Precio total (€)**
D-Sub	Conector D-Sub Combinado, DC-21WA4	1	11,44	11,44
BNC	Conector de RF/Coaxial, Coaxial BNC, Macho Recto, Soldable, 50 ohm	3	4,82	14,46
TRMNL	Virola para Cable, Terminación DIN, Cable Simple, 20 AWG, 0.5 mm2, 8 mm, Blanco	100	0,043	4,3
CCXL	Cable Coaxial, 0.14 mm2, 50 ohm, 30.5 m	1	30,53	30,53
CMPT	Cable Multipar, Blindado, 1 Par, 22 AWG	1	1,03	1,03
			TOTAL	61,76

Impresión 3D y Tornillería				
ID	**Descripción**	**Ud**	**Precio Ud (€)**	**Precio total (€)**
TT1	Tornillo, M3, 10 mm, Cabeza Cilíndrica y Plana Ranurada	1	2,37	2,37
TT2	Tornillo, M3, 30 mm, A2, Ranurada con Cabeza	1	7,62	7,62
TT3	Tuerca, M3, Hexagonal	1	1,34	1,34
PLA1	Filamento de Impresora 3D, 1.75mm, 1Kg, PLA, Blanco	1	38,18	38,18
			TOTAL	49,51

17 Mano de obra

Mano de Obra			
Tipo de mano de obra	**Unidades (h)**	**Honorarios(€/h)**	**TOTAL (€)**
Graduado en Ingeniería Electrónica Industrial y Automática	337	40	13480,00

18 Coste total

TOTAL	
Concepto	**Coste(€)**
Pasarelas WIZ750SR-110	79,04
Optoacopladores Láseres AVIA y Spirit	18,59
DAQ_Arduino	88,35
Cableado	61,76
Impresión3D y Tornillería	49,51
Mano de Obra	13480,00
TOTAL	13777,25
TOTAL (+21 % IVA)	16670,47

El presupuesto total asciende a la cantidad de:

#Dieciséis mil seiscientos setenta con cuareta y siete euros (16.670,47€)#